

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

MICROSOFT CORPORATION  
Petitioner,

v.

EDGE NETWORKING SYSTEMS, LLC,  
Patent Owner.

---

U.S. Patent No. 11,695,823  
Issue Date: July 4, 2023  
Title: DISTRIBUTED SOFTWARE DEFINED NETWORKING

---

*Inter Partes* Review No.: Unassigned

---

**DECLARATION OF EREZ ZADOK, PH.D.  
ON BEHALF OF PETITIONER**

***Mail Stop "PATENT BOARD"***  
Patent Trial and Appeal Board  
U.S. Patent and Trademark Office  
P.O. Box 1450  
Alexandria, VA 22313-1450

## TABLE OF CONTENTS

	<b>Page</b>
I. Introduction.....	8
II. Background and Qualifications .....	10
III. Materials Considered.....	21
IV. Understanding of Relevant Legal Principles.....	29
A. Claim Construction .....	29
B. Obviousness.....	32
C. Ordinary Skill in the Art.....	36
V. Person of Ordinary Skill in the Art.....	37
VI. Technology Overview .....	38
A. General Computer Operations.....	39
1. Networking Overview.....	41
B. Data Security and Privacy .....	46
1. Hash Functions and Their Uses .....	50
C. Virtualization.....	53
D. Operating Systems and Sandboxing.....	60
1. Operating Systems and Processes.....	61
2. Hypervisors and Virtual Machines .....	65
3. Java Virtual Machines and Java Applications .....	65
E. Security for Software Implemented and Downloaded over the Internet Including Digital Signature Implementations.....	67
F. Distributed Systems.....	76

1.	Open Standards Gateway Initiative (OSGi).....	88
2.	Clouds .....	106
VII.	The '823 Patent.....	114
A.	Summary of the '823 Patent.....	115
B.	Prosecution History of the '823 Patent .....	134
C.	Priority Date of the Challenged Claims .....	143
VIII.	Overview of the Grounds for Challenge .....	143
A.	Overview of the Prior Art in the Grounds for Challenge.....	144
1.	Vasell (EX1004/EX1015/EX1016) .....	145
2.	The Alves Textbook (EX1008).....	169
3.	The Hall Textbook (EX1009) .....	175
4.	Rellermeyer (EX1011).....	180
B.	Claim Construction .....	183
1.	“virtual fabric” (in claim 12): .....	185
2.	“load controller adapted to monitor loads ... and affect change in accordance with thresholds received ...” (claim 7).....	186
IX.	Challenges: Claims 1-2, 12-15 and 19 of the '823 Patent are Obvious over Vasell in view of Alves and Rellermeyer, and claims 3-5, 7-8 and 18 of the '823 Patent are Obvious over Vasell in view of Alves, Rellermeyer and Hall.....	187
A.	Motivation to Combine the Teachings of the Prior Art References .....	187
B.	Claim Element by Claim Element Invalidity Analysis: '823 Patent Claims 1-2, 12-15, and 19 are Obvious Over Vasell in View of Alves and Rellermeyer. ....	197
1.	Claim 1 .....	197

[1.0]	A system comprising:.....	197
[1.1]	a programmable network device adapted to host a plurality of network device applications;.....	202
[1.2]	a programmable cloud device adapted to host a plurality of cloud applications, .....	205
[1.3]	wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form a distributed application; .....	214
[1.4]	wherein the plurality of network device applications and plurality of cloud applications device form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device application and plurality of cloud applications independent of network device hardware and cloud device hardware. ....	227
2.	Claim 2: The system of claim 1, wherein the programmable network device and the programmable cloud device each include an access network interface unit configured to send and receive communications and a processor with a memory associated with the networ interface unit.....	233
7.	Claim 19 .....	247
[19.0]	A system comprising:.....	247
[19.1]	a programmable network device adapted to host a plurality of network device applications;.....	248
[19.2]	a programmable cloud device adapted to host a plurality of cloud applications .....	248

	[19.3] wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications;.....	248
	[19.4] wherein the plurality of network device applications and plurality of cloud applications device form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device application and plurality of cloud applications independent of network device hardware and cloud device hardware.....	248
	[19.5] an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications; and.....	249
	[19.6] an infrastructure application marketplace in communication with the application management portal, said infrastructure application marketplace capable of providing tested distributed applications to the application management portal. ....	252
C.	Claim Element by Claim Element Invalidity Analysis: '823 Patent Claims 3-5, 7-8, and 18 are Obvious Over Vasell in View of Alves, Rellermeyer and Hall. ....	259
1.	Claim 3:.....	259
	[3.0] The system of claim 1 further comprising: .....	259
	[3.1] an application management portal capable of managing life cycles of the distributed applications; and.....	259
	[3.2] an infrastructure application marketplace in communication with the application	

	management portal, said infrastructure marketplace capable of providing tested and certified distributed applications to the application management portal. ....	259
2.	Claim 4: The system of claim 3, wherein the application management portal is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications.....	264
3.	Claim 5: The system of claim 3, further comprising: a distributed resource service (DRS) which controls access to a plurality of resources upon receiving instructions from the application management portal. ....	266
4.	Claim 7: The system of claim 5, wherein the DRS further includes a load controller adapted to monitor loads on at least one of the plurality of network device applications and at least one of the plurality of cloud applications and effect change in accordance with thresholds received from the application management portal.....	271
5.	Claim 8: The system of claim 3 further comprising: a distributed notification service wherein notice of a predetermined event is sent from at least one of the plurality of network device applications to at least one of the plurality of cloud applications .....	279
6.	Claim 18: The system of claim 1, wherein the at least one programmable network device and the programmable cloud device are adapted to verify the authenticity and integrity of updates to the plurality of network device applications and the plurality of cloud applications.....	281
X.	Conclusion .....	294
a.	Availability for Cross-Examination .....	294
b.	Right to Supplement .....	294

c. Jurat.....294

I, Erez Zadok, Ph.D., declare as follows:

## **I. INTRODUCTION**

1. I am over the age of eighteen (18) and otherwise competent to make this Declaration.

2. I have been retained by Microsoft Corporation (“Petitioner”) as an independent, technical expert consultant in this proceeding before the United States Patent and Trademark Office (“USPTO”). My consulting company, Zadoks Consulting, LLC, is being compensated for my time at my current standard consulting rate. No part of my compensation is dependent on my opinions or on the outcome of this proceeding. I have no financial interest in any of the parties to this proceeding.

3. This Declaration is in support of a petition for *inter partes* review involving U.S. Patent No. 11,695,823 (the “’823 Patent” or “the Patent”) (EX1001), entitled “Distributed Software Defined Networking” and listing Pouya Taaghhol and Vivek Ramanna as the inventors. The ’823 Patent issued July 4, 2023 from U.S. Patent Application No. 17/142,983, which was filed on January 6, 2021, and which claimed priority to U.S. Pat. No. 10,893,095, which was filed on June 14, 2020, and also to U.S. Patent No. 10,686,871, which was filed on December 9, 2017, U.S. Patent No. 9,843,624, which was filed on June 4, 2014, and U.S. Provisional Patent Application No. 61/834,807, which was filed on June 13, 2013. EX1001 (’823

Patent), 1-2.

4. I have been informed and understand that, according to USPTO records, the '823 Patent is currently assigned to Edge Networking Systems, LLC ("Edge Networking Systems" or "Patent Owner"). I assume for purposes of this proceeding that Edge Networking Systems is the patent owner.

5. For the purposes of this *inter partes* review proceeding, as I discuss later (in §VII.C below), I have been instructed to assume that the effective filing date of the claims of the '823 Patent challenged by the Petitioner in this *inter partes* review proceeding is the earliest identified priority date, *i.e.*, the June 13, 2013 filing date of U.S. Provisional Patent Application No. 61/834,807.

6. In preparing this Declaration, I have reviewed the '823 Patent (EX1001) and its prosecution history (EX1002), as well as the patents and documents cited herein. I have considered these documents in light of the general knowledge in the art as of June 13, 2013. In formulating my opinions, I have relied upon my experience in the relevant art. I have also considered the viewpoint of a person of ordinary skill in the art ("POSITA") in the relevant field, as of June 13, 2013, who I describe in §V below. For convenience, the materials I considered in arriving at my opinions that I express in this Declaration are listed in §III below (Materials Considered).

## II. BACKGROUND AND QUALIFICATIONS

7. In formulating my opinions, I have relied on my knowledge, training, and experience in the relevant field, which I will summarize briefly. A more detailed summary of my background, education, experience, and publications is set forth in my curriculum vitae (CV), which is submitted as EX1035.

8. I have personal knowledge of the facts and opinions set forth in this Declaration and believe them to be true. If called upon to do so, I would testify competently thereto. I have been warned that willful false statements and the like are punishable by fine or imprisonment, or both. I have had no contact with any of the named inventors of the '823 Patent.

9. My consulting company, Zadoks Consulting, LLC, is being compensated for my time at my current standard consulting rate. I am also being reimbursed for expenses that I may incur during the course of this work. My compensation is not contingent upon the results of my study and analysis, the substance of my opinions, or the outcome of any proceeding involving the '823 Patent. I have no financial interest in the outcome of this matter or in any litigation involving the '823 Patent.

10. I am a Professor in the Computer Science Department at Stony Brook University (part of the State University of New York ("SUNY") system). I direct the File-systems and Storage Lab (FSL) at Stony Brook's Computer Science

Department. My research interests include file systems and storage systems, operating systems, transactional systems including database technologies, information technology and system administration, security/privacy and information assurance, networking, energy efficiency, performance and benchmarking, virtualization, cloud systems, compilers, applied machine learning, and software engineering.

11. I studied at a professional high school in Israel, focusing on electrical engineering (“EE”), and graduated in 1982. I spent one more year at the high school’s college division, receiving a special Certified Technician’s degree in EE. I then went on to serve in the Israeli Defense Forces for three years (1983-1986). I received my Bachelor of Science degree in computer science (“CS”) in 1991, my Master’s degree in CS in 1994, and my PhD in CS in 2001—all from Columbia University in New York.

12. When I began my undergraduate studies at Columbia University, I also started working as a student assistant in the various campus-wide computer labs, eventually becoming an assistant to the head labs manager, who was managing all public computer labs on campus. During that time, I also became more involved with research within the CS Department at Columbia University, conducting research on operating systems, file and storage systems, distributed and networked systems, security, and other topics. I also assisted the CS department’s computer

administrators in managing the department's computers, which included storage, IT, networking, and cyber-security related duties.

13. In 1991, I joined Columbia University's CS department as a full-time systems administrator, studying towards my MS degree part-time. My MS thesis topic is related to file system reliability, fault tolerance, replication, and failover in mobile networked storage systems using file virtualization. My main duties as a systems administrator involved installing, configuring, and managing many networked servers, proxies, and desktops running several operating systems, as well as network devices setup; this included many software and hardware upgrades, device upgrades, and BIOS firmware/chipset updates/upgrades. My duties also included ensuring reliable, secure, authenticated access to networked systems/storage and licensed software, as well as software updates, security and bug fixes. Examples of servers and their protocols included email (SMTP), file transfer (FTP), domain names (DNS), network file systems (NFS), network news systems (NNTP), and Web (HTTP).

14. In 1994, I left my systems administrator position to pursue my doctoral studies at Columbia University. My PhD thesis topic was on versatile file system development using stackable (virtualized) file systems, with examples in the fields of security and encryption, efficiency, reliability, and failover. I continued to work part-time as a systems administrator at the CS department, and eventually I was

asked to serve as manager to the entire information technology (“IT”) staff. From 1991 to 2001, I was also a member of the faculty-level Facilities Committee that oversaw all IT operations at the CS department.

15. As part of my PhD studies at Columbia, I collaborated on projects to develop advanced AI-like techniques to detect previously unknown viruses (a.k.a. “zero-day malware”), using data mining and rule-based detection. This work led to several highly cited papers (over 1,600 citations for one of the papers alone), and two patents. I also became a Teaching Assistant (TA) for a first-ever Computer Security course given at Columbia University’s CS department with Dr. Matt Blaze as instructor.

16. From 1990 to 1998, I consulted for SOS Corporation and HydraWEB Technologies, as a systems administrator and programmer, managing data storage use and backup/restore duties, databases, web servers, as well as information assurance and cyber-security (*e.g.*, malware protection, software licensing). From 1994 to 2000, I led projects at HydraWEB Technologies, and then became the Director of Software Development—overseeing the development of several products and appliances such as stateful firewalls and HTTP load-balancers, utilizing network-virtualization and high-availability techniques. From 2009 to 2019, I consulted for Packet General Networks, a startup specializing in secure, virtualized, network storage and applications’ data security in the cloud.

17. In 2001, I joined the faculty of Stony Brook University, a position I have held since that time. In 2002, I joined the Operations Committee, which oversees the IT operations of the CS department at Stony Brook University. From 2006 to 2010, I was the Director of IT Operations of the CS department. My day-to-day duties included setting policies regarding computing, hiring and training new staff, assisting any staff with topics of my specialty, defining requirements for new software/hardware, and purchasing. From 2010 to 2015, I had served as the Co-Chair to the Operations Committee. From 2016 to 2019, I oversaw the IT Operations as the Chair of the Operations Committee. A significant component of these duties included defining and helping implement policies for data management, to ensure the security of users and their data, and data reliability and availability, while minimizing the inconvenience and performance impact to users. I personally helped setup and maintain an initial virtual-host infrastructure in the department. Since late 2019, I have been a member of the department's Executive Committee that also oversees all IT operations.

18. In 2017, I became the department's Graduate Academic Adviser, advising all Master students (over 400 annually on average) and many other graduate students on an assortment of academic matters. In August 2024, I took over as the department's Graduate Program Director, overseeing the entire graduate CS program (700-800 students annually on average).

19. Since 2001, I have personally configured and managed my own research lab's network. This includes setting up and configuring multiple storage systems (e.g., NFS, CIFS/SMB, NAS), virtual and physical environments, applications such as database (e.g., MySQL, Postgresql), Web servers (e.g., Apache), and mail servers; user access control (e.g., NIS, LDAP), backups and restores, snapshot policies, and more. I have personally installed, configured, changed, replaced parts, and upgraded components in numerous devices including mobile devices, laptops, desktops, and servers, both physical and virtual.

20. Since 1995, I have taught courses on operating systems, storage and file systems, advanced systems programming in Unix/C, systems administration, data structures, data/software security, and more. My courses often use storage, file systems, distributed systems, and system/network security as key teaching principles and practical examples for assignments and projects. I have taught these concepts and techniques to my students, both to my direct advisees as well as in my courses. For example, in my graduate Operating Systems course, I often cover Linux's kernel mechanisms to protect users, applications, and data files, virtual file systems, as well as distributed storage systems (e.g., NFS). And in the System Administration undergraduate course, I covered many topics such as networking, storage, backups, and configuring complex applications such as mail, web, and database servers.

21. My research often investigates computer systems from many angles:

security, efficiency, energy use, scalability, reliability, portability, survivability, usability, ease-of-use, versatility, flexibility, and more. My research gives special attention to balancing five often-conflicting aspects of computer systems: performance, reliability, energy use, security, and ease-of-use.

22. Since joining Stony Brook University in 2001, my group in the File- systems and Storage Lab (FSL) has developed many file systems and operating system extensions; examples include a highly-secure cryptographic file system, a portable copy-on-write (COW) versioning file system, a tracing file system useful to detect intrusions, a replaying file system useful for forensics, a snapshotting and sandboxing file system, a namespace unification file system (that uses stackable, virtualized, file-based COW), an anti-virus file system, an integrity-checking file system, a load balancing and replication/mirroring file system, network file system extensions for security and performance, distributed secure cloud-based storage systems, transactional key-value stores and file systems, OS-level embedded databases, a compiler to convert user-level C code to in-kernel efficient yet safe code, GCC plugins, stackable file system templates, and a Web-based backup system. Many of these projects used one form of virtualization or another (storage, network, host, etc.). I continue to maintain and release newer versions of some of these file systems and software.

23. I have published over 120 refereed publications (in ACM, IEEE,

USENIX, and more). To date, my publications have been cited more than 10,000 times (as per Google Scholar as of September 24, 2024). My papers cover a wide range of related technologies such as file systems, storage systems, transactional systems, security, clouds and virtualization, performance benchmarking and optimization, energy efficiency, system administration, web systems, and more. I also published a book titled “Linux NFS and Automounter Administration” (Sybex, 2001), covering systems administration topics related to network storage and data security.

24. Some of my research has led to public software releases that have been used worldwide. I have publicly maintained the Amd Berkeley Automounter in a package called “am-utils” since 1992; this software helps administrators manage the multitude of file system mounts on dozens of different Unix systems, especially helping to automate access to multiple NFS/NAS storage volumes. Since 1997, I have maintained and released several stackable (virtualized) file system software projects for Linux, FreeBSD, and/or Sun Solaris, in a package called FiST. One of my stackable file system encryption projects, called Cryptfs, became the basis for IBM’s public release of eCryptfs, now part of Linux. Packet General Networks, for whom I have provided consulting services between 2009 and 2019, licensed another encryption file system called Ncryptfs. Another popular file system released in 2003, called Unionfs, offers virtual namespace unification, transparent

shadow copying (a.k.a. copy-on-write or COW), file system snapshotting (*e.g.*, useful for forensics and disaster recovery), and the ability to save disk space by sharing a read-only copy of data among several computers, among other features.

25. My research and teaching make extensive use of data security features. For example, each time I taught the graduate operating system course, the first homework assignment includes the creation of a new system call that performs new or added functionality, often for encrypting a file or verifying its integrity; many of my other assignments cover topics of user/process access control, anti-virus filtering, and more. Since 2001, over 1,000 graduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

26. Moreover, in an undergraduate course titled “Advanced Systems Programming in Unix/C,” I cover many topics of system security and vulnerabilities, such as the structure of UNIX processes, and memory segments such as the heap and stack. This course covers details of several hundred Linux system calls. Often, the first assignment for this course is to develop a tool to encrypt/decrypt files using advanced ciphers, use digital signatures to certify the cipher keys used, and reliably recover files in case of failures. Since 2001, several hundred undergraduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

27. In another undergraduate course, System Administration, I taught

network configuration, security, and storage configuration and reliability. In a special topics course on Storage Systems, I covered many topics such as data deduplication, RAID, transactional storage, storage hardware including modern Flash based ones, virtual storage, backup/restore, snapshots and continuous data protection (CDP), NAS and SAN, and NFS.

28. Overall, in addition to the aforementioned experience, my technical experience relevant to the '823 Patent at the time of its effective filing date (June 13, 2013) included the following: I configured, used, researched, programmed, and used several cloud systems and virtualized environments; I taught, researched, programmed, and used dozens of operating systems, processes, and drivers; I configured and programmed sandboxing systems including for anti-malware detection, for running critical system services (*e.g.*, FTP, web, and email servers); I configured and used virtual networks; I used, taught, and programmed several data security system including using public-private keys and hashes to verify software installations on many systems I managed; and I configured and wrote software to monitor other computer systems and software, including control its performance through thresholds, rate-controls, and load-balancing.

29. My research has been supported by many federal and state grants as well as industry awards, including an NSF CAREER award, two IBM Faculty awards, two NetApp Faculty awards, a Western Digital award, a Facebook award,

several Dell-EMC awards, and several equipment gifts. I received the 2008 SUNY Chancellor's Excellence in Teaching award, and the 2022 SUNY Chancellor's Award for Excellence in Scholarship and Creative Activities (both awards can be given only once in a lifetime). In 2021, I was named an ACM Distinguished Member for "Outstanding Scientific Contributions to Computing."

30. My service record to the community includes serving as the co-chair for the USENIX Annual Technical Conference in 2020 (ATC'20); serving as the co-chair for USENIX File and Storage Technologies (FAST'15) in 2015 and on the FAST Conference Steering Committee from 2015 to 2023; serving on the ACM HotStorage Steering Committee since 2021; and serving as the co-chair in 2012 and on the Steering Committee of the ACM SYSTOR conference since 2012. I have served as an Associate Editor to the ACM Transactions on Storage (TOS) journal from 2009 to 2022; in 2022, I was named the Editor-in-Chief for ACM's TOS journal.

31. I am a named inventor on four patents, two titled "Systems and Methods for Detection of New Malicious Executables" (U.S. Patent No. 7,487,544, issued February 3, 2009; and U.S. Patent No. 7,979,907, issued July 12, 2011); and two more titled "Multi-Tier Caching," (U.S. Patent No. 9,355,109, issued May 31, 2016; and U.S. Patent 9,959,279, issued May 1, 2018).

32. I have been disclosed as a testifying expert in 25 cases (including *inter*

*partes* review (IPR) proceedings) in the past four years. I have been deposed 13 times and testified in trial twice.

### III. MATERIALS CONSIDERED

33. In forming my opinions expressed in this declaration, I reviewed the following materials:

Exhibit #	Description
1001	U.S. Patent No. 11,695,823 (the “’823 Patent”).
1002	Prosecution history of the ’823 Patent (the “’823 File History”).
1004	U.S. Patent No. 6,496,575 to Vasell, et al. (“Vasell Patent”) (collectively, with EX1015 and EX1016, “Vasell”).
1005	File history of the parent patent to the ’823 Patent, U.S. Patent No. 9,843,624 (the “’624 Patent”).
1006	File history of the patent from which the ’823 Patent was filed as a continuation application, U.S. Patent No. 10,893,095 (the “’095 Patent”).
1007	File history of the other patent from which the ’823 Patent claims priority in a chain of continuation applications, U.S. Patent No. 10,686,871 (the “’871 Patent”).
1008	Excerpts from A. de Castro Alves, “OSGi in Depth”, Manning Publications Company, 2012 (“Alves”).
1009	Excerpts from R. Hall, et al., “OSGi in Action - Creating Modular Applications in Java”, Manning Publications Company, April 2011 (“Hall”).
1010	J. E. Kim, et al., “Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes,” 2012 Eighth International Conference on Intelligent Environments, Guanajuato, Mexico, June 26-29, 2012, pp. 206-213 (“Kim”).
1011	J. S. Rellermeyer and S. Bagchi, “Dependability as a cloud service - a modular approach,” IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012), Boston, MA, USA, 2012, pp. 1-6 (“Rellermeyer”).

Exhibit #	Description
1012	S. Kächele, et al., “Component-based scalability for cloud applications”, Proceedings of the 3 <sup>rd</sup> International Workshop on Cloud Data and Platforms (CloudDP ‘13), April 14, 2013, Prague, Czech Republic, 19–24 (“Kächele”).
1013	OSGi Alliance, “About the OSGi Service Platform”, Technical Whitepaper, Revision 4.1, November 11, 2005, archived on November 30, 2005 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20051130032628/http://www.osgi.org/documents/collateral/TechnicalWhitePaper2005osgi-spooverview.pdf">https://web.archive.org/web/20051130032628/http://www.osgi.org/documents/collateral/TechnicalWhitePaper2005osgi-spooverview.pdf</a> (“OSGi 2005 Whitepaper”).
1014	Declaration of June A. Munford (“Munford Librarian Declaration”).
1015	U.S. Provisional Patent Application No. 60/088,437, filed on June 8, 1998, claimed as a priority application in, and incorporated by reference into, EX1004 (“Vasell 1 <sup>st</sup> Provisional”) (collectively, with EX1004 and EX1016, “Vasell”).
1016	U.S. Provisional Patent Application No. 60/123,971, filed on March 12, 1999, claimed as a priority application in, and incorporated by reference into, EX1004 (“Vasell 2 <sup>nd</sup> Provisional”) (collectively, with EX1004 and EX1015, “Vasell”), which includes a copy of the article published as Idermark, T., Lilliestråle, M., & Vasell, J., “Ericsson’s e-box system—An electronic services enabler”, Ericsson Review, (1), 1999, 38-44, as confirmed by Ericsson’s website archived, for example, on March 3, 2000 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20000303195310/http://www.ericsson.se/review/issues.taf">https://web.archive.org/web/20000303195310/http://www.ericsson.se/review/issues.taf</a> and, for example, on February 16, 2003 and November 27, 2004 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20030216191220/http://www.ericsson.com/about/publications/review/1999_01/23.shtml">https://web.archive.org/web/20030216191220/http://www.ericsson.com/about/publications/review/1999_01/23.shtml</a> and <a href="https://web.archive.org/web/20041127035720/http://www.ericsson.com/about/publications/review/1999_01/files/1999015.pdf">https://web.archive.org/web/20041127035720/http://www.ericsson.com/about/publications/review/1999_01/files/1999015.pdf</a> respectively.
1017	“About Gatespace”, Gatespace AB website, archived on January 6, 2000 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20000106145021/http://www.gatespace.com/">https://web.archive.org/web/20000106145021/http://www.gatespace.com/</a> (“About Gatespace 2000”).

Exhibit #	Description
1018	<p>“Carlstedt Research &amp; Technology Forms New Company for Developing Software for ‘The Intelligent Home’ in Cooperation with Ericsson”, September 30, 1999, Gatespace AB website, archived on February 12, 2001 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20010212054449/http://www.gatespace.com/news/archive/19990930_en_v3.shtml">https://web.archive.org/web/20010212054449/http://www.gatespace.com/news/archive/19990930_en_v3.shtml</a> (“Gatespace Formation Press Release”).</p>
1019	<p>“Ericsson e-services is a part of the Open Services Gateway Initiative (OSGI)”, Ericsson e-services website, archived on February 29, 2000 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20000229111550/http://www.ericsson.com:80/wireless/products/ebox/links/osgi.shtml">https://web.archive.org/web/20000229111550/http://www.ericsson.com:80/wireless/products/ebox/links/osgi.shtml</a> (“Ericsson E-Services WebPage”).</p>
1020	<p>“Fifteen industry leaders to create standard for bringing Internet-based services to the networked home”, March 1, 1999, Ericsson e-services website, archived on May 28, 2000 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20000528223608/http://www.ericsson.com/wireless/products/ebox/news/990301.shtml">https://web.archive.org/web/20000528223608/http://www.ericsson.com/wireless/products/ebox/news/990301.shtml</a> (“OSGi Announcement on Ericsson E-Services WebPage”).</p>
1021	<p>“People”, Carlstedt Research &amp; Technology AB website, archived on January 6, 2000 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20000106204945/http://www.crt.se/about/people.en.html">https://web.archive.org/web/20000106204945/http://www.crt.se/about/people.en.html</a> (“CR&amp;T People 2000”).</p>
1022	<p>“Open Services Gateway Initiative: Charter”, OSGi website, archived on April 19, 2000 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20000419181219/http://www.osgi.org/about/charter.html">https://web.archive.org/web/20000419181219/http://www.osgi.org/about/charter.html</a> (“OSGi Charter 2000”).</p>
1023	<p>“Open Services Gateway Initiative: Officers”, OSGi website, archived on April 14, 2000 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20000414133426/http://www.osgi.org/about/officers.html">https://web.archive.org/web/20000414133426/http://www.osgi.org/about/officers.html</a> (“OSGi Officers 2000”).</p>
1024	<p>“Open Services Gateway Initiative Elects New Leadership”, Home Toys Inc. website, May 23, 2001, archived on July 8, 2001 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20010708123711/http://www.hometoys.com/releases/jun01/osgi01.htm">https://web.archive.org/web/20010708123711/http://www.hometoys.com/releases/jun01/osgi01.htm</a> (“OSGi Leadership 2001”).</p>

Exhibit #	Description
1025	“Management Team”, Gatespace AB website, archived on February 8, 2002 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20020208031748/http://www.gatespace.com/company/management.shtml">https://web.archive.org/web/20020208031748/http://www.gatespace.com/company/management.shtml</a> (“Gatespace Management 2002”).
1026	“About Us: Background”, Gatespace AB website, archived on February 6, 2002 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20020206205742/http://www.gatespace.com/company/">https://web.archive.org/web/20020206205742/http://www.gatespace.com/company/</a> (“About Gatespace 2002”).
1027	“About Makewave: History in Making”, Makewave AB website, archived on August 16, 2007 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20070816163615/http://www.makewave.com/site.en/about/history.shtml">https://web.archive.org/web/20070816163615/http://www.makewave.com/site.en/about/history.shtml</a> (“About Makewave 2007”).
1028	The OSGi Alliance, “OSGi Service Platform Enterprise Specification”, Release 4, Version 4.2, March 2010 (“OSGi Enterprise Specification”).
1029	T. Forst, “Cisco Application eXtension Platform – OSGi Add-on as universal Middleware for Your Applications”, May 2008, AutomatedBuildings.com, archived on May 17, 2008 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20080517111602/www.automatedbuildings.com/news/may08/articles/prosyst/080427034829prosyst.htm">https://web.archive.org/web/20080517111602/www.automatedbuildings.com/news/may08/articles/prosyst/080427034829prosyst.htm</a> (“Forst”).
1030	U.S. Patent Publication No. 2013/0086147 to Kashyap, filed October 3, 2011, published April 4, 2013 (“Kashyap”).
1031	Excerpts from A. S. Tanenbaum, “Modern Operating Systems”, Prentice Hall, 3 <sup>rd</sup> edition, 2008 (“Tanenbaum 2008”).
1032	Excerpts from A. S. Tanenbaum, “Computer Networks”, Prentice Hall, 3 <sup>rd</sup> edition, 1996 (“Tanenbaum 1996”).
1033	Excerpts from D. Downing, et al., “Dictionary of Computer and Internet Terms”, Barron’s Educational Series, Inc., 11 <sup>th</sup> edition, 2013 (“Downing”).
1034	Excerpts from E. Cole, et al., “Network Security Bible”, Wiley Publishing, Inc., 2 <sup>nd</sup> edition, 2009 (“Cole”).
1035	My Curriculum Vitae (CV).
1036	S. Soltesz, et al., “Container-based Operating System Virtualization: A Scalable, High Performance Alternative to Hypervisors”, In Proceedings of the 2 <sup>nd</sup> ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys ‘07), March 21-23, 2007, Lisbon, Portugal, pp. 275-287, 2007 (“Soltesz”).

Exhibit #	Description
1037	<i>Intentionally Left Blank</i>
1038	<i>Intentionally Left Blank</i>
1039	Li, et al., “ExpoNet: A Flexible Platform for Concurrent Experiments on Programmable Infrastructure”, 2011 IEEE Global Telecommunications Conference - GLOBECOM 2011, Houston, TX, USA, 2011, pp. 1-5 (“Li”).
1040	<i>Intentionally Left Blank</i>
1041	P. Kriens, “The Bundle Repository”, OSGi Alliance Blog, April 7, 2006, archived on May 4, 2006 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20060504203641/http://www.osgi.org/blog/2006/04/bundle-repository.html">https://web.archive.org/web/20060504203641/http://www.osgi.org/blog/2006/04/bundle-repository.html</a> (“Kriens OSGi Blog”).
1042	“Knoplerfish Pro Premium” and “Knoplerfish Pro Enterprise” Data Sheets, Makewave AB, 2010, respectively archived on February 20, 2011 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20110220123804/http://www.makewave.com:80/resources/docs/datasheets/MD-042-A1-knoplerfish_pro_premium_osgi.pdf">https://web.archive.org/web/20110220123804/http://www.makewave.com:80/resources/docs/datasheets/MD-042-A1-knoplerfish_pro_premium_osgi.pdf</a> and <a href="https://web.archive.org/web/20110220123819/http://www.makewave.com:80/resources/docs/datasheets/MD-047-A1-knoplerfish_pro_enterprise_osgi.pdf">https://web.archive.org/web/20110220123819/http://www.makewave.com:80/resources/docs/datasheets/MD-047-A1-knoplerfish_pro_enterprise_osgi.pdf</a> (“Knoplerfish”).
1043	R. Kawashima, “vNFC: A Virtual Networking Function Container for SDN-enabled Virtual Networks”, 2012 Second Symposium on Network Cloud Computing and Applications, London, UK, 2012, pp. 124-129 (“Kawashima”).
1044	Excerpts from A. S. Tanenbaum, “Modern Operating Systems”, Prentice Hall, 2 <sup>nd</sup> edition, 2001 (“Tanenbaum 2001”).
1045	Excerpts from S. Garfinkel, et al., “Web Security & Commerce”, O’Reilly & Associates, 1997 (“Garfinkel”).

Exhibit #	Description
1046	<p>Oasis Standard Specification, “Web Services Security: SOAP Message Security 1.1 (WS-Security 2004), February 1, 2006, archived on February 6, 2007 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20070206143722/http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-soapmessagesecurity.pdf">https://web.archive.org/web/20070206143722/http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-soapmessagesecurity.pdf</a>, and accessible from OASIS Web Services Security (WSS) Technical Committee web page, as archived on December 5, 2006 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20061205032600/http://oasis-open.org/committees/tc_home.php?wg_abbrev=wss#technical">https://web.archive.org/web/20061205032600/http://oasis-open.org/committees/tc_home.php?wg_abbrev=wss#technical</a> (“Oasis - SOAP Security”).</p>
1047	<p>“Web Service Security Cheat Sheet”, The Open Web Application Security Project (OWASP), archived on June 12, 2012 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20120612205800/https://www.owasp.org/index.php/Web_Service_Security_Cheat_Sheet">https://web.archive.org/web/20120612205800/https://www.owasp.org/index.php/Web_Service_Security_Cheat_Sheet</a>, and accessible from OWASP’s (the free and open software security community) web pages, as archived on June 10, 2012 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20120610001928/https://www.owasp.org/index.php/Main_Page">https://web.archive.org/web/20120610001928/https://www.owasp.org/index.php/Main_Page</a>, and as archived on June 10, 2012 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20120610091415/https://www.owasp.org/index.php/Cheat_Sheets">https://web.archive.org/web/20120610091415/https://www.owasp.org/index.php/Cheat_Sheets</a> (“OWASP - WSS Security”).</p>
1048	<p>“REST Security Cheat Sheet”, The Open Web Application Security Project (OWASP), archived on January 15, 2013 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20130115190935/https://www.owasp.org/index.php/REST_Security_Cheat_Sheet">https://web.archive.org/web/20130115190935/https://www.owasp.org/index.php/REST_Security_Cheat_Sheet</a>, and accessible from OWASP’s (the free and open software security community) web pages, as archived on January 4, 2013 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20130104230717/https://www.owasp.org/index.php/Main_Page">https://web.archive.org/web/20130104230717/https://www.owasp.org/index.php/Main_Page</a>, and as archived on January 15, 2013 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20130115215253/https://www.owasp.org/index.php/Cheat_Sheets">https://web.archive.org/web/20130115215253/https://www.owasp.org/index.php/Cheat_Sheets</a> (“OWASP – REST Security”).</p>
1049	<i>Intentionally Left Blank</i>

<b>Exhibit #</b>	<b>Description</b>
1050	The Open Services Gateway Initiative, “Specification Overview”, Version 1.0, January 2000, archived on August 31, 2000 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20000831061202/http://www.osgi.org/about/specoverview.pdf">https://web.archive.org/web/20000831061202/http://www.osgi.org/about/specoverview.pdf</a> (“OSGi 2000 Overview”).
1051	R. Hall, “Oscar Bundle Repository”, archived on June 30, 2004 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20040630163519/http://oscar-osgi.sourceforge.net:80/">https://web.archive.org/web/20040630163519/http://oscar-osgi.sourceforge.net:80/</a> (“OBR 2004”).
1052	Excerpts from B. Sosinsky, “Cloud Computing Bible”, Wiley Publishing, Inc., 2011 (“Sosinsky”).
1053	“Java Secure Socket Extension (JSSE) Reference Guide for Java Platform Standard Edition 7”, Oracle Java SE Documentation, archived on November 25, 2011 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20111125044243/http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html">https://web.archive.org/web/20111125044243/http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html</a> (“JSSE Guide”).
1054	“Java™ Remote Method Invocation API (Java RMI)”, Oracle Java SE Documentation, archived on November 27, 2011 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20111127072151/http://docs.oracle.com/javase/7/docs/technotes/guides/rmi/index.html">https://web.archive.org/web/20111127072151/http://docs.oracle.com/javase/7/docs/technotes/guides/rmi/index.html</a> (“Java RMI Overview”).
1055	“Using Custom Socket Factories with Java RMI”, Oracle Java SE Documentation, archived on April 3, 2012 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20120403095031/https://docs.oracle.com/javase/7/docs/technotes/guides/rmi/socketfactory/index.html">https://web.archive.org/web/20120403095031/https://docs.oracle.com/javase/7/docs/technotes/guides/rmi/socketfactory/index.html</a> (“Java RMI Socket Overview”).
1056	“Using Java™ RMI with SSL”, Oracle Java SE Documentation, archived on July 3, 2012 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20120703221253/https://docs.oracle.com/javase/7/docs/technotes/guides/rmi/socketfactory/SSLInfo.html">https://web.archive.org/web/20120703221253/https://docs.oracle.com/javase/7/docs/technotes/guides/rmi/socketfactory/SSLInfo.html</a> (“Java RMI SSL Overview”).
1057	Microsoft Computer Dictionary, 3rd ed., 1997, excerpts (“Microsoft Computer Dictionary”).

Exhibit #	Description
1058	Usenix Jails in FreeBSD for Fun and Profit, Paco Hope, USENIX ;login: The Magazine of Usenix & Sage, vol. 27, number 3, June 2002 (“Hope”), archived on March 16, 2003 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20030316225604/http://www.usenix.org/publications/login/2002-06/pdfs/hope.pdf">https://web.archive.org/web/20030316225604/http://www.usenix.org/publications/login/2002-06/pdfs/hope.pdf</a> .
1059	Excerpts from Silberschatz et al., Operating System Concepts (8 <sup>th</sup> Ed. 2009) (“Silberschatz 2009”).
1060	Excerpts from Silberschatz et al., Operating System Concepts (9 <sup>th</sup> Ed. 2012) (“Silberschatz 2012”).
1061	Excerpts from McKusick and Neville-Neil, “The Design and Implementation of the FreeBSD Operating System” (2005) (“McKusick”).
1062	Excerpts from Benevenuti, “Understanding Linux Network Internals” (2006) (“Benevenuti”).
1063	Excerpts from Stallings, “Operating Systems Internals and Design Principles” (7 <sup>th</sup> ed. 2012) (“Stallings”).
1064	“apache_1.3.19.tar.gz.md5” Hash File, Apache Software Foundation, archived on June 28, 2001 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20010628043950/http://www.apache.org/dist/httpd/apache_1.3.19.tar.gz.md5">https://web.archive.org/web/20010628043950/http://www.apache.org/dist/httpd/apache_1.3.19.tar.gz.md5</a> .
1065	“apache_1.3.19.tar.gz.asc” Signature File, Apache Software Foundation, archived on June 28, 2001 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20010628043146/apache.org/dist/httpd/apache_1.3.19.tar.gz.asc">https://web.archive.org/web/20010628043146/apache.org/dist/httpd/apache_1.3.19.tar.gz.asc</a> .
1066	“Apache Archive Distribution Directory”, Apache Software Foundation, available at <a href="https://archive.apache.org/dist/httpd">https://archive.apache.org/dist/httpd</a> (“Apache Archive”).
1067	U.S. Department of Commerce, FIPS PUB 186-2, Digital Signature Standard (DSS) (January 27, 2000) (“DSS FIPS PUB”).
1068	J. Preshing, “Hash Collision Probabilities”, Preshing on Programming, May 4, 2011, archived on September 27, 2011 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20110927200414/preshing.com/20110504/hash-collision-probabilities">https://web.archive.org/web/20110927200414/preshing.com/20110504/hash-collision-probabilities</a> (“Preshing Hash Collision Probabilities”).
1069	“Avalanche Effect”, Wikipedia, archived on April 30, 2010 at Internet Archive’s Wayback Machine at <a href="https://web.archive.org/web/20100430122132/https://en.wikipedia.org/wiki/Avalanche_effect">https://web.archive.org/web/20100430122132/https://en.wikipedia.org/wiki/Avalanche_effect</a> (“Wikipedia Avalanche Effect”).

<b>Exhibit #</b>	<b>Description</b>
1070	Plaintiff Edge Networking Systems, LLC’s Opening Claim Construction Brief, filed as Document No. 31, on December 17, 2024, in Case No. 1:24-cv-00215-DAE, captioned as <i>Edge Networking Systems, LLC v. Microsoft Corporation</i> (W.D. Texas).
1071	Defendant Microsoft Corporation’s Responsive Claim Construction Brief, filed as Document No. 32, on January 17, 2025, in Case No. 1:24-cv-00215-DAE, captioned as <i>Edge Networking Systems, LLC v. Microsoft Corporation</i> (W.D. Texas).

#### IV. UNDERSTANDING OF RELEVANT LEGAL PRINCIPLES

34. I am not an attorney. For the purposes of this declaration, I have been informed about certain aspects of the law that are relevant to my opinions. My understanding of the law is as follows:

##### A. Claim Construction

35. I have been informed by counsel and I understand that claim construction is a matter of law and that the final claim construction will ultimately be determined by the Board. For the purposes of my analysis in this proceeding and with respect to the prior art, I have been informed by counsel that IPRs are currently reviewed under “the *Phillips* standard.”

36. I have been informed by counsel and I understand that, under the *Phillips* standard, claim terms are given their plain and ordinary meaning as understood by a person of ordinary skill in the art at the time of the invention in light of the claim language and the patent specification.

37. I have been informed by counsel and I understand that embodiments

described in the specification are generally encompassed by the claims.

38. I have been informed by counsel and I understand that the patentee can serve as their own lexicographer. As such, if a claim term is provided with a specific definition in the specification, then I should interpret that claim term in light of the particular definition provided by the patentee.

39. I have been informed by counsel and I understand that claim elements may be expressed as a means for performing a recited function as set out in 35 U.S.C. § 112(f). I have also been informed by counsel and I understand that one or more terms within a patent claim may be written in means-plus-function form, even if the claim term does not include the specific word “means.” I have further been informed and I understand that if a claim element does not contain the term “means,” it is presumed not to be a means-plus-function limitation that is subject to 35 U.S.C. § 112(f). However, I have been informed and I understand that this presumption can be rebutted by showing that the claim element recites a function without reciting sufficient structure for performing that function, from the perspective of a POSITA.

40. I have been informed by counsel and I understand that certain terms have been explicitly recognized by courts as “nonce” words, which I understand to be verbal constructs that are not recognized as the name of structure and are effectively a substitute for the term “means for.” I have been informed by counsel and I understand that the proper inquiry is whether the words of the claim itself,

when read in light of the specification, connote to a POSITA definite structure for performing the identified functions.

41. I have also been informed by counsel and understand that for such a means-plus-function element, the element is to be construed to cover the corresponding structure, material, or acts described in the patent specification for performing that function and equivalents thereof. Thus, I understand that, in the case of a means-plus-function element, the scope of the claim element is limited to only structure that is both actually disclosed in the patent specification and clearly linked to the claimed function(s).

42. I have been informed by counsel and I understand that a claim including a means-plus-function element is literally disclosed if the prior art is found to have a structure that performs the identical recited function wherein that structure is identical or equivalent to structure disclosed in the patent for performing the identical recited function. I further understand that structures are deemed equivalent if they are insubstantially different. One way of determining whether structures are equivalent is to determine whether each performs the identical recited function in a substantially similar way to obtain a substantially similar result. I understand that a structural equivalence analysis must be supported by specific evidence, and that a conclusory statement alleging that a structure within prior art is equivalent to structure disclosed in the patent for performing the identical recited function is

insufficient.

43. I also have been informed by counsel and I understand that when construing means-plus-function limitations that concern a computer or a microprocessor that is programmed to carry out an algorithm, the structure is to be construed as the algorithm as disclosed in the patent specification. I further understand that disclosure of a means-plus-function claim limitation directed to a computer programmed to perform an algorithm requires that the software in the prior art uses an algorithm that performs the same steps as the algorithm disclosed in the patent specification.

**B. Obviousness**

44. I have been informed by counsel and I understand that a patent claim can be considered to have been obvious to a person of ordinary skill in the art at the time the application was filed. This means that, even if all of the requirements of a claim are not found in a single prior art reference, the claim is not patentable if the differences between the subject matter in the prior art and the subject matter in the claim would have been obvious to a person of ordinary skill in the art at the time the application was filed (here June 13, 2013).

45. I have been informed by counsel and I understand that a determination of whether a claim would have been obvious should be based upon several factors, including, among others:

- the level of ordinary skill in the art at the time the application was filed;
- the scope and content of the prior art; and
- what differences, if any, existed between the claimed invention and the prior art.

46. I have been informed and I understand that a claim can be unpatentable in light of a single prior art reference if it would have been obvious to modify that reference to come up with the claimed subject matter.

47. I have been informed and understand that the teachings of two or more references may be combined in the same way as disclosed in the claims, if such a combination would have been obvious to one having ordinary skill in the art.

48. I have been informed by counsel and I understand that whether there are any relevant differences between the prior art and the claimed invention is to be analyzed from the view of a person of ordinary skill in the relevant art at the time the application was filed. As such, my opinions below as to a person of ordinary skill in the art are as of the time the application was filed, even if not expressly stated as such; for example, even if stated in the present tense.

49. In analyzing the relevance of the differences between the claimed invention and the prior art, I have been informed by counsel and I understand that I must consider the impact, if any, of such differences on the obviousness or non-obviousness of the invention as a whole, not merely some portion of it. The person

of ordinary skill in the art faced with a problem is able to apply his or her experience and ability to solve the problem and also look to any available prior art to help solve the problem.

50. I have also been informed by counsel and I understand that a precise teaching in the prior art directed to the subject matter of the claimed invention is not needed. I have been informed by counsel and I understand that one may take into account the inferences and creative steps that a person of ordinary skill in the art would have employed in reviewing the prior art at the time of the invention. For example, if the claimed invention combined elements known in the prior art and the combination yielded results that were predictable to a person of ordinary skill in the art at the time of the invention, then this evidence would make it more likely that the claim was obvious. On the other hand, if the combination of known elements yielded unexpected or unpredictable results, or if the prior art teaches away from combining the known elements, then this evidence would make it more likely that the claim that successfully combined those elements was not obvious.

51. In determining whether a combination based on either the teachings of a single reference or the teachings of multiple references would have been obvious, I have been informed by counsel and I understand that it is appropriate to consider, among other factors:

- whether the teachings of the prior art references disclose known concepts

combined in familiar ways, which, when combined, would yield predictable results;

- whether a person of ordinary skill in the art would have implemented a predictable variation, and would have seen the benefit of doing so;
- whether the claimed elements represent one of a limited number of known design choices, and would have a reasonable expectation of success by those skilled in the art;
- whether a person of ordinary skill would have recognized a reason to combine known elements in the manner described in the claim;
- whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent; and
- whether the claim applies a known technique that had been used to improve a similar device or method in a similar way.

52. I understand that one of ordinary skill in the art has ordinary creativity and is not an automaton.

53. I understand that in considering obviousness, it is important not to determine obviousness using the benefit of hindsight derived from the patent being considered.

54. I have been informed by counsel and I understand that objective indicia can be important evidence regarding whether a patent claim is obvious or

nonobvious, if it has an appropriate nexus to the claimed invention, *i.e.*, if it is a result of the merits of a claimed invention (rather than the result of design needs or market-pressure advertising or similar activities). Such indicia include: (a) commercial success of products covered by the patent claims; (b) a long-felt need for the invention; (c) failed attempts by others to make the invention; (d) copying of the invention by others in the field; (e) unexpected results achieved by the invention as compared to the closest prior art; (f) praise of the invention by the infringer or others in the field; (g) the taking of licenses under the patent by others; (h) expressions of surprise by experts and those skilled in the art at the making of the invention; (i) the patentee proceeded contrary to the accepted wisdom of the prior art; and, (j) the contemporaneous development of the subject matter claimed by others.

**C. Ordinary Skill in the Art**

55. I have been informed by counsel and I understand that, in the context of an invalidity analysis, a person having ordinary skill in the art (“POSITA”) is a hypothetical person who is presumed to be aware of all pertinent prior art, thinks along conventional wisdom in the art, and is a person of ordinary creativity.

56. I have also been informed by counsel and understand that there are multiple factors relevant to determining the level of ordinary skill in the pertinent art, including (1) the levels of education and experience of persons working in the

field at the time of the invention; (2) the sophistication of the technology; (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems. I understand that these factors need not all be taken into account for the analysis and that one or more of these factors may control.

## **V. PERSON OF ORDINARY SKILL IN THE ART**

57. As I discussed in §IV.C above (Understanding of Relevant Legal Principles – Ordinary Skill in the Art), I have been informed by counsel and understand that a person of ordinary skill in the relevant art (“POSITA”) is a hypothetical person who is presumed to be aware of all pertinent prior art, thinks along conventional wisdom in the art, and is a person of ordinary creativity. I have also been informed by counsel and understand that there are multiple factors relevant to determining the level of ordinary skill in the pertinent art, including (1) the levels of education and experience of persons working in the field at the time of the invention; (2) the sophistication of the technology; (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems.

58. In my opinion, a person of ordinary skill in the art (“POSITA”) in the field of the ’823 Patent as of its June 13, 2013 Priority Date (*see* §VII.C below) would have had a bachelor’s degree in computer science, computer engineering, or equivalent degree, and approximately three years of experience working in the computer science or engineering field. Additional experience might substitute for

less education and vice versa. To that end, POSITAs in June 2013 would have been knowledgeable about the design and management of networked systems and virtualization technologies, and familiar with operating/distributed systems and security and privacy techniques.

59. I met these requirements for a POSITA at the time of the June 13, 2013 Priority Date of the '823 Patent. *See* §II above (Background and Qualifications).

## VI. TECHNOLOGY OVERVIEW

60. In this section, I provide an overview of the background art as understood by a person of ordinary skill in the art as of June 13, 2013.

61. Throughout my Declaration, I use the color scheme identified in the below table to annotate the same elements in the '823 Patent (EX1001) and the prior art references that I discuss herein:

Concept	Color
distributed system for processing data packets comprising a network device and a cloud device	alternating <b>green</b> and <b>maroon</b>
network device	<b>green</b>
cloud device	<b>maroon</b>
distributed application with a component on a network device and a component on a cloud device	alternating <b>pink</b> and <b>orange</b>
network device application that is a component of the distributed application hosted on a network device	<b>pink</b>

cloud application that is a component of the distributed application hosted on a cloud device	orange
application management portal (and associated software)	blue
application repository/store	brown

**A. General Computer Operations**

62. Computers are hardware and software devices that execute instructions.

The hardware often includes a central processing unit (CPU) or other microprocessors, typically called “processor,” volatile memory (*e.g.*, RAM (random-access memory), or DRAM (dynamic random-access memory)), non-volatile (persistent) storage (*e.g.*, a hard disk or flash drive), and input/output (I/O) devices (*e.g.*, network interface card, mouse, keyboard, display). RAM, hard disks, and flash drives are considered, generally, “storage” or “memory” devices because they can store and hold bits of information (whether persistently or not). Computers can have multiple storage devices. Processors are coupled to main memory using a memory bus and to peripherals such as hard disks and network interface cards using one or more I/O busses.

63. Memory and storage devices have considerable differences in their speeds, capacities, and costs. Non-volatile storage devices such as hard disks have the largest capacity, are least expensive per gigabyte, and are slowest (typically operating at millisecond speeds). Volatile DRAM devices are smaller, more

expensive, and faster than hard disks (typically operating at microsecond speeds). CPUs' own memory caches and operations are the smallest, most expensive, and fastest (typically operating at nanosecond speeds). Overall, CPUs are about 1,000× faster than DRAM, which in turn is about 1,000× faster than hard disks.

64. Computer programs are often written in a high-level human-readable language (*e.g.*, C, C++, Java, Perl, PHP, Python), then translated using a compiler or script processor to machine instructions understood by the CPU (*e.g.*, Intel or AMD processor). Machine instructions—generally called “software”—are stored in files on persistent media (*e.g.*, HDD or ROM), loaded into DRAM, and then loaded into the CPU where they can be executed. All modern computers operate in this manner, regardless of their purpose: a small laptop, a server in a data center, a mainframe computer, a personal workstation, a proxy device, a gateway, a firewall, a router, an appliance, a virtual machine, a handheld or mobile device, smart card, SD card, etc. Computer systems may run multiple software programs, from background servers and daemons to user applications.

65. Users (including administrators) can login to one or more computers (*e.g.*, desktop, laptop, server) and execute one or more applications (*e.g.*, word processing, web browsing, backup tool). Some computers offer external interfaces and APIs to communicate with certain devices and especially peripherals. For example, both a USB storage device as well as a printer use a well-defined

“protocol” to allow users to communicate with them through the main computer.

## 1. Networking Overview

66. A Local Area Network (LAN) is a typically a network at a local site where network latencies are short and physical distances between computers are small. A LAN and its computers are often protected from the rest of the internet using a firewall. EX1057 (Microsoft Computer Dictionary) at 276 (“**LAN** *n.* Acronym for **local area network**. A group of computers and other devices dispersed over a relatively limited area and connected by a communications link that enables any device to interact with any other on the network. LANs commonly include microcomputers and shared resources such as laser printers and large hard disks. The devices on a LAN are known as nodes, and the nodes are connected by cables through which messages are transmitted. *See also* baseband network, broadband network, bus network, collision detection, communications protocol, contention, CSMA/CD, network, ring network, star network, token bus network, token passing, token ring network. *Compare* wide area network.”).

67. Another form of a local area network is one that operates inside a single computer. A “localhost” is “[t]he name that is used to represent the same computer on which a TCP/IP message originates. An IP packet sent to localhost has the IP address 127.0.0.1 and does not actually go out to the Internet. *See also* IP address, packet (definition 1), TCP/IP.” EX1057 (Microsoft Computer Dictionary) at 287.

Often, when users/processes want to communicate with other users/processes, using TCP/IP communications on the same machine, they can use the localhost address: it is considered a trusted, high-speed internal network.

68. Conversely, a Wide-Area Network (WAN) refers to computers across the entire Internet, who are distant from each other, exhibit longer latencies, and afford little protection. EX1057 (Microsoft Computer Dictionary) at 507 (“**wide area network** *n.* A communications network that connects geographically separated areas. *Acronym:* WAN.”).

69. A firewall typically intercepts all traffic between computers in a LAN and the WAN, monitors and controls access, and can filter out undesired traffic. Often, LAN computers are far more trusted than WAN computers; therefore, a firewall can be said to separate the trusted network (LAN or localhost) from the untrusted network (WAN). EX1057 (Microsoft Computer Dictionary) at 197 (“**firewall** *n.* A security system intended to protect an organization’s network against external threats, such as hackers, coming from another network, such as the Internet. A firewall prevents computers in the organization’s network from communicating directly with computers external to the network and vice versa. Instead, all communication is routed through a proxy server outside of the organization’s network, and the proxy server decides whether it is safe to let a particular message or file pass through to the organization’s network.”).

70. A server is a computer or software that provides a service to other computers, called clients. There are many servers on the Internet, for example, Web servers and Mail servers. EX1057 (Microsoft Computer Dictionary) at 430 (“**server** *n.* 1. On a local area network (LAN), a computer running administrative software that controls access to the network and its resources, such as printers and disk drives, and provides resources to computers functioning as workstations on the network. 2. On the Internet or other network, a computer or program that responds to commands from a client. For example, a file server may contain an archive of data or program files; when a client submits a request for a file, the server transfers a copy of the file to the client. *See also* client/server architecture. *Compare* client (definition 3).”

71. A client is typically a computer or software, sometimes used by a user, that communicates across LAN and WAN networks; a client may contact servers to request some service. A user’s desktop, laptop, and handheld device are examples of clients. An end-point computer is an example of a client, but note that a computer can be both a client and a server (*e.g.*, a proxy device), and that a server could be referred to as a client (*e.g.*, in distributed systems, *see* §VI.F below), depending on its role. EX1057 (Microsoft Computer Dictionary) at 92 (“**client** *n.* 1. In object-oriented programming, a member of a class (group) that uses the services of another class to which it is not related. *See also* inheritance (definition 1). 2. A process, such as a program or task, that requests a service provided by another program—for

example, a word processor that calls on a sort routine built into another program. The client process uses the requested service without having to ‘know’ any working details about the other program or the service itself. *Compare* child (definition 1), descendant (definition 2). 3. On a local area network or the Internet, a computer that accesses shared network resources provided by another computer (called a server). *See also* client/server architecture, server.”).

72. A client/server architecture describes how client computers interact with server computers. EX1057 (Microsoft Computer Dictionary) at 92 (“**client/server architecture** *n.* An arrangement used on local area networks that makes use of distributed intelligence to treat both the server and the individual workstations as intelligent, programmable devices, thus exploiting the full computing power of each. This is done by splitting the processing of an application between two distinct components: a ‘front-end’ client and a ‘back-end’ server. The client component is a complete, stand-alone personal computer (not a ‘dumb’ terminal), and it offers the user its full range of power and features for running applications. The server component can be a personal computer, a minicomputer, or a mainframe that provides the traditional strengths offered by minicomputers and mainframes in a time-sharing environment: data management, information sharing between clients, and sophisticated network administration and security features. The client and server machines work together to accomplish the processing of the

application being used. Not only does this increase the processing power available over older architectures but it also uses that power more efficiently. The client portion of the application is typically optimized for user interaction, whereas the server portion provides the centralized, multiuser functionality. *See also* distributed intelligence.”).

73. A proxy device or server, such as a firewall or caching server, also acts as a client. A “proxy” or “proxy server” is “[a] firewall component that manages Internet traffic to and from a local area network (LAN) and can provide other features, such as document caching and access control. A proxy server can improve performance by supplying frequently requested data, such as a popular Web page, and can filter and discard requests that the owner does not consider appropriate, such as requests for unauthorized access to proprietary files. *See also* firewall.” EX1057 (Microsoft Computer Dictionary) at 387.

74. A “data packet” or “packet” is “1. A unit of information transmitted as a whole from one device to another on a network. 2. In packet-switching networks, a transmission unit of fixed maximum size that consists of binary digits representing both data and a header containing an identification number, source and destination addresses, and sometimes error-control data. *See also* packet switching.” EX1057 (Microsoft Computer Dictionary) at pp. 132, 348.

## **B. Data Security and Privacy**

75. All modern systems include measures of security to protect their data from unauthorized access. EX1057 (Microsoft Computer Dictionary), p. 195 (“**file protection** *n.* A process or device by which the existence and integrity of a file are maintained. Methods of file protection range from allowing read-only access and assigning passwords to covering the write-protect notch on a disk and locking away floppy disks holding sensitive files.”).

76. The POSIX standard, first started in the late 1980s, codifies the access application program interfaces (APIs) that operating systems export to user applications. EX1057 (Microsoft Computer Dictionary), p. 351 (“**POSIX** *n.* Acronym for **P**ortable **O**perating **S**ystem **I**nterface for **U**NIX. An IEEE standard that defines a set of operating-system services. Programs that adhere to the POSIX standard can be easily ported from one system to another. POSIX was based on UNIX system services, but it was created in a way that allows it to be implemented by other operating systems.”).

77. A simple data protection scheme defines whether users can only read files, only write files, or have both read and write access. EX1057 (Microsoft Computer Dictionary), p. 361 (“**permission** *n.* In a networked or multiuser computer environment, the ability of a particular user to access a particular resource by means of his or her user account. Permissions are granted by the system administrator or

other authorized person; these permissions are stored in the system (often in a file called a *permissions log*) and are checked when a user attempts to access a resource.”).

78. In the basic POSIX model, for example, a file has three sets of permissions (read, write, or execute) for three classes of users (owner, group member, and everyone else).

79. More sophisticated schemes generalize these simpler permission models to allow for lists of users, lists of groups, as well as logical operations on lists (*e.g.*, negation, conjunction, and disjunction). These are called “access control lists,” or ACLs. EX1057 (Microsoft Computer Dictionary), p. 13 (“**access control** *n.* The mechanisms for limiting access to certain items of information or to certain controls based on users’ identities and their membership in various predefined groups. Access control is typically used by system administrators for controlling user access to network resources, such as servers, directories, and files. *See also* access privileges, system administrator.”), p. 13 (“**access control list** *n.* A list associated with a file that contains information about which users or groups have permission to access or modify the file. *Acronym:* ACL.”). ACLs existed in Unix systems since at least the 1990s (and I personally used them).

80. Another common technique, useful to protect user’s privacy, is “encryption”, or “[t]he process of encoding data to prevent unauthorized access,

especially during transmission. Encryption is usually based on a key that is essential for decoding. The U.S. National Bureau of Standards created a complex encryption standard, Data Encryption Standard (DES), which provides almost unlimited ways to encrypt documents. *See also* DES.” EX1057 (Microsoft Computer Dictionary) at 175.

81. There are many forms of verification that can be applied to data files. One example is to verify the integrity of data using cryptographic hashing algorithms such as SHA. “SHA” is an “[a]cronym for **S**ecure **H**ash **A**lgorithm. A technique that computes a 160-bit condensed representation of a message or data file, called a *message digest*. The SHA is used by the sender and the receiver of a message in computing and verifying a digital signature, for security purposes. *See also* algorithm, digital signature.” EX1057 (Microsoft Computer Dictionary) at 432. The term “data integrity” represents “[t]he accuracy of data and its conformity to its expected value, especially after being transmitted or processed.” EX1057 (Microsoft Computer Dictionary) at 131.

82. Users can also verify the authenticity of files’ sources using digital signatures. A “digital signature” is “[a] personal authentication method based on encryption and secret authorization codes used for ‘signing’ electronic documents.” EX1057 (Microsoft Computer Dictionary) at 145.

83. Digital signatures often combine cryptographic techniques and hashing

algorithms to enable bi-directional authentication and secure communications. “Public key cryptography” or “public key encryption” is “[a]n asymmetric scheme that uses a pair of keys for encryption: the public key encrypts data, and a corresponding secret key decrypts it. For digital signatures, the process is reversed: the sender uses the secret key to create a unique electronic number that can be read by anyone possessing the corresponding public key, which verifies that the message is truly from the sender. *See also* private key, public key.” EX1057 (Microsoft Computer Dictionary) at 388-389.

84. Digital signatures have been used to authenticate software being distributed, including Java. *See* §VI.D.3 below. A POSITA would know that distributing many binaries safely on the Internet often included the dissemination of separate hash and digital signature files that the user/software downloading the binary can use to verify the download.

85. For example, the Apache Software Foundation had distributed its popular Web server at least as far back as 2003, as separate compressed “tarball” files (*e.g.*, `apache_1.3.19.tar.gz`) along with a separate MD5 hash files (*e.g.*, `apache_1.3.19.tar.gz.md5` whose contents include a one line 128-bit MD5 hash encoded as 32 hexadecimal characters) (EX1064) and separate digital signature files (*e.g.*, `apache_1.3.19.tar.gz.asc` whose contents include a lengthy multi-line signature generated by the PGP 6.5.1i tool) (EX1065).

86. A POSITA would understand that, if the `apache_1.3.19.tar.gz` tarball file were moved or copied to a different web site (no different than copying/moving JAR files to different web sites), then the tarball's separate MD5 hash would remain valid and uniquely identify that tarball; however, the digital signature may no longer be valid on a different web site and may have to be re-signed with the new web site's identity or certificate. *See* EX1066 (Apache Archive) at 7.

87. In fact, digital signatures were so common that in 2000 the U.S. Government published a standard describing their use. *See* EX1067 (a document published by the U.S. Department of Commerce and the National Institute of Standards and Technology titled "Digital Signature Standard (DSS)," dated January 27, 2000).

### **1. Hash Functions and Their Uses**

88. Hashing algorithms are mathematical functions that take as input (typically) a long piece of data and produce a shorter output number that can (almost always) uniquely identify the input data. Depending on the specific context, hash functions are sometimes called checksums, fingerprints, message digests, or digital signature functions.

89. Hash functions are one-way: it is typically considered cryptographically hard to produce the original input data from the hash itself (*e.g.*, invert it). *See* EX1057 (Microsoft Computer Dictionary) at pp. 228 ("**hash**<sup>2</sup> *vb.* To

be mapped to a numerical value by a transformation known as a hashing function. Hashing is used to convert an identifier or key, meaningful to a user, into a value for the location of the corresponding data in a structure, such as a table”), 432 (“**SHA** *n.* Acronym for **S**ecure **H**ash **A**lgorithm. A technique that computes a 160-bit condensed representation of a message or data file, called a *message digest*.”), 122-123 (“**CRC** *n.* Acronym for **c**yclical (or **c**yclic) **r**edundancy **c**heck.... CRC error checking uses a complex calculation to generate a number based on the data transmitted.”).

90. Hashing functions are said to produce a “probabilistically unique” output because there is a small chance that two different inputs may produce the same output hash value. The mathematical probability itself depends on the length of the hash value and the number of items being hashed: the smaller the hash and the more items are hashed, the higher this “collision” probability is. Weaker hash functions often use 32- to 64-bit hashes (*e.g.*, CRC32, MD4); stronger hashes, or cryptographically strong ones, often produce hashes that are at least 128-160 bits long (*e.g.*, MD5, SHA1). *See* EX1068 (Preshing Hash Collision Probabilities).

91. Well-designed hash functions are supposed to exhibit the “avalanche effect”: small input changes should produce large output changes. *See* EX1069 (Wikipedia Avalanche Effect). If a single input bit changed (*e.g.*, flipped), the avalanche effect calls for half of the output hash bits to have changed on average.

This means that it should be rare that few or most bits would have changed due to a single input bit change. This property helps ensure the (probabilistic) uniqueness of hash values, their uniformity across the full possible space, and the minimization of collisions.

92. Hash functions have been used for many years in various settings from security, cryptography, integrity, data deduplication, and others. For example, the Tripwire system I personally used in the early 1990's hashes files and their meta-data and compares those hashes against the files later on; if a mismatch is detected, it means that the file was changed, which could indicate that an intruder broke into the system and installed malware such as Trojans. *See also* §II above (Background and Qualifications).

93. In another example, vendors hash known malicious binaries and distribute those to customers. The hashes are small, can easily be distributed, and also do not identify the actual binary (because customers do not like exposing their own files even to their own security vendors). On the customer's computers, existing and incoming binaries can be compared: any matches found would indicate that the customer has or received malware. This malware can be cleaned (if possible), quarantined (*e.g.*, saved in a special folder), deleted, and/or reported.

94. An alternative to handle the zero-day malware is to detect the malicious program by its behavior rather than a hash of the file's contents or parts thereof. For

example, a program that reads an entire computer's password file and immediately appears to transmit it over the network is more likely to be malware. (Such behavioral scanning or activity monitoring to detect previously unknown malware using its behavior was the subject of my own work in the late 1990's, which produced several highly cited and award-winning publications, as well as two patents, as I described in §II above (Background and Qualifications).)

95. Hashes are useful beyond security applications. Storage systems and file systems can store numerous files. Many of those files, or parts thereof, can be duplicates (*e.g.*, due to users copying files repeatedly, backup procedures, and more). Storage systems therefore have employed data de-duplication techniques to remove these duplicates: this can save a lot of space and reduce costs. Because files' names do not have to be unique and can change over time, deduplication systems often use a different identifier for the file's actual data—unique hash computed from the actual data (or parts thereof). In that way, the hash can uniquely identify the file's data regardless of the file's name or location.

### **C. Virtualization**

96. Among computer scientists, there is an axiom stating that “you can solve almost any problem by adding another level of indirection.” Indeed, virtualization is an indirection technique that makes one thing seem like another; virtualization often presents a physical-world artifact as one that appears as an

illusion or “fake”. For something to be “virtual”, it is “[o]f or pertaining to a device, service, or sensory input that is perceived to be what it is not in actuality, usually as more ‘real’ or concrete than it actually is.” EX1057 (Microsoft Computer Dictionary), p. 497.

97. Indeed, the Microsoft Computer Dictionary includes 35 examples and definitions of virtualization. EX1057 (Microsoft Computer Dictionary), pp. 497-500.

98. In one example, a “virtual machine” is “[s]oftware that mimics the performance of a hardware device, such as a program that allows applications written for an Intel processor to be run on a Motorola chip. *Acronym: VM.*” EX1057 (Microsoft Computer Dictionary), p. 498.

99. In another example, “virtual memory” is “[m]emory that appears to an application to be larger and more uniform than it is. Virtual memory may be partially simulated by secondary storage such as a hard disk. Applications access memory through virtual addresses, are translated (mapped) by special hardware and software onto physical addresses. *Acronym: VM. Also called disk memory. See also paging, segmentation.*” EX1057 (Microsoft Computer Dictionary), p. 498.

100. In yet another example, “virtual reality” is “[a] simulated 3-D environment that a user can experience and manipulate as if it were physical. The user sees the environment on display screens, possibly mounted in a special pair of

goggles. Special input devices, such as gloves or suits fitted with motion sensors, detect the user's actions. *Acronym: VR.*" EX1057 (Microsoft Computer Dictionary), p. 499.

101. Virtualization had been applied to storage systems, disk devices, and drivers. A "virtual device driver" is "[s]oftware in Windows 95 that manages a hardware or software system resource. If a resource retains information from one access to the next that affects the way it behaves when accessed (for example, a disk controller with its status information and buffers), a virtual device driver must exist for it." EX1057 (Microsoft Computer Dictionary), p. 498.

102. A "virtual disk" or "RAM disk" is "[s]hort for random access memory disk. A simulated disk drive whose data is actually stored in RAM memory. A special program allows the operating system to read from and write to the simulated device as if it were a disk drive. RAM disks are extremely fast, but they require that system memory be given up for their use. Also, RAM disks usually use volatile memory, so the data stored on them disappears when power is turned off. Many portables offer RAM disks that use battery-backed CMOS RAM to avoid this problem. *See also* CMOS RAM. *Compare* disk cache." EX1057 (Microsoft Computer Dictionary), pp. 395-396, 498.

103. Virtualization is sometimes said to be "logical" representation of physical reality. "Logical" is "[o]f or pertaining to a conceptual piece of equipment

or frame of reference, regardless of how it may be realized physically. *Compare* physical.” EX1057 (Microsoft Computer Dictionary), p. 288 (“logical” definition 2).

104. A “logical drive” or “logical device” is “[a] device named by the logic of a software system, regardless of its physical relationship to the system. For example, a single floppy disk drive can simultaneously be, to the MS-DOS operating system, both logical drive A and drive B.” EX1057 (Microsoft Computer Dictionary), p. 288.

105. Moreover, one can virtualize files and file systems. This was the subject of much of my graduate-level research and became part of my PhD Dissertation. *See* §II above (Background and Qualifications) and EX1035 (Zadok Curriculum Vitae).

106. Virtualization, of course, has been applied to networking as well. A “virtual network” is “[a] part of a network that appears to a user to be a network of its own. For example, an Internet service provider can set up multiple domains on a single HTTP server so that each one can be addressed with its company’s registered domain name. *See also* domain name, HTTP server (definition 1), ISP.” EX1057 (Microsoft Computer Dictionary), p. 499.

107. A “virtual LAN” is “[s]hort for **virtual local area network**. A local area network consisting of groups of hosts that are on physically different segments but

that communicate as though they were on the same wire. *See also* LAN.” EX1057 (Microsoft Computed Dictionary), p. 498.

108. One of the most famous uses of a (secure) virtual network is a private one. A “virtual private network” (VPN) is “[a] set of nodes on a public network such as the Internet that communicate among themselves using encryption technology so that their messages are as safe from being intercepted and understood by unauthorized users as if the nodes were connected by private lines. 2. A wide area network formed of permanent virtual circuits (PVCs) on another network, especially a network using technologies such as ATM or frame relay. *Acronym:* VPN. *See also* ATM (definition 1), frame relay, PVC.” EX1057 (Microsoft Computed Dictionary), p. 499. A POSITA would have understood definition 2 of VPN to be a form of an overlay network because it is a collection of individual virtual network segments.

109. Virtualized networking has been popular, for example in Linux systems. “VLAN stands for Virtual LAN. The use of VLANs is a convenient way to isolate traffic using the same L2 switch in different broadcast domains by means of an additional tag, called the VLAN tag, that is added to the Ethernet frames.” EX1062 (Benevenuti), p. 43 (footnote 2).

110. Linux supports “Virtual IP” mechanisms. EX1062 (Benevenuti), p. 704. “Another common use for gratuitous ARP is to allow failover in a pool of

servers. Commonly, to provide redundancy, a site provides one active server along with a number of similarly configured hosts in standby mode. When the active server fails for some reason, a mechanism often referred to as a *heartbeat timer* (implemented through some protocol on the pool of servers) detects the failure and triggers the election of a new active server. This new server generates a gratuitous ARP to update the ARP cache of all the other hosts in the network. Because the new server has taken the PI address of the old server, the ARPOP\_REQUEST is not answered, but all the recipients update their caches accordingly. Note that in this way, the IP layer and higher layers can keep communicating without even noticing the change.” EX1062 (Benevenuti), p. 704. That is, by updating the Ethernet MAC address (via GARP), the now-virtual IP address of a host becomes the active one.

111. As of July 2012, “[v]irtual machine technology, often just called virtualization,... [was] more than 40 years old”, was well-known to “allow[] a single computer to host multiple virtual machines, each potentially running a different operating system”, and had well-known advantages (e.g., “strong isolation” including “failure in one virtual machine does not automatically bring down any others”, permitted “compan[ies]... [with] hundreds of thousands of servers doing a huge variety of different tasks” to “hav[e] fewer physical machines... represent[ing] a huge costs savings”, “checkpointing and migrating virtual machines (e.g., for load balancing across multiple servers) is... eas[y]”, simplified “software development”,

etc.) and tradeoffs (although “most service outages are not due to faulty hardware, but due to bloated, unreliable buggy software, especially operating systems”, “**if** the server [hardware] running all the virtual machines fails, the result is even more catastrophic than a single dedicated server crashing.”). EX1031 (Tanenbaum 2008), 569-570 (emphasis added in bold).

112. As another example of a single computer (*e.g.*, server) hosting multiple virtual machines, Soltesz (EX1036) describes “container-based operating system (COS) technology” (example depicted below), “hosted on a shared OS image” (Linux kernel) and a “privileged host VM” (Linux VServer, “the VM that a system administrator uses to manage other VMs”), including a plurality of virtual machines (each hosting a plurality of applications), where (i) “[e]ach VM can be booted, shut down, and rebooted just like a regular operating system”; (ii) “[r]esources such as disk space, CPU guarantees, memory, etc. are assigned to each VM when it is created, yet often can be dynamically varied at run time”; and (iii) “[t]o applications and the user of a container-based system, the VM appears just like a separate host”:

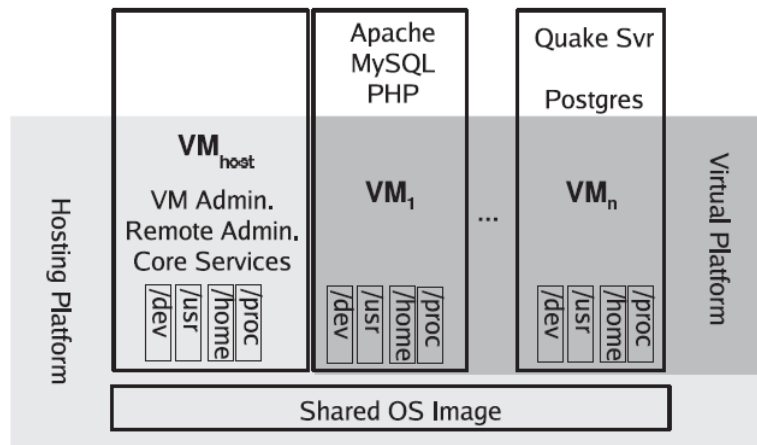


Figure 3: COS Overview

EX1036 (Soltesz), 278-279, Figure 3 (reproduced above).

#### D. Operating Systems and Sandboxing

113. The ‘823 Patent describes a “sandboxing operating system” as a form of isolating, containing, or confining some software from another. “The sandboxing operating system ensures each application runs as a dedicated process [to] in sure isolation from the other applications. The fxOS 302a may be built based on an existing sandboxed operating system (OS) (e.g., Android®) by extending several aspects of such an OS with routing/networking layers and supports for operating as the wireless infrastructure.” EX1001 (’823 Patent), 10:15-21; *see also* §VII.A below (Summary of the ’823 Patent).

114. A POSITA would have understood this to not be limited to traditional operating systems such as Android (which itself is based on Unix/Linux). As I describe next, software/process isolation, confinement, containment, and/or

sandboxing occur in a variety of forms. The common thread to this sandboxing is that there is always some layer logically below the isolated entities that enforces said isolation. Examples include (1) an operating system isolating processes and containers from each other (and the OS itself); (2) a hypervisor isolating whole virtual machines (VMs) from each other; and (3) the Java Virtual Machine (JVM), isolating Java programs from each other.

### **1. Operating Systems and Processes**

115. “The processes in an operating system must be protected from one another’s activities. To provide such protection, we can use various mechanisms to ensure that only processes that have gained proper authorization from the operating system can operate on the files, memory segments, CPU, and other resources of a system.” EX1059 (Silberschatz 2009), p. 591.

116. Traditional operating systems include mechanisms to sandbox applications (*e.g.*, confining and restricting what resources an application can access and when).

117. One of the earliest examples in Unix OSs was using the change-root (*chroot*) system call. “A *chroot* environment is one in which a process’s view of the file system is restricted to a specific part of the hierarchy. The process’s file system has a virtual root. Probably the most readily available example of a commonly *chrooted* process is the FTP daemon. ... FTP is not the only process commonly

chrooted these days. IMAP software and DNS software are both commonly chrooted as well. The chroot solution is attractive for protecting certain kinds of vulnerable processes. It is usually used to guard against software that can be coerced to read or write files that it should not touch in normal operations. Such vulnerabilities are limited in scope to just the chroot area of the file system. Thus the files available to such a vulnerable process are dramatically reduced.” EX1058 (Hope), p. 2.

118. Another form of sandboxing is for whole operating systems via virtualization. “Virtual operating systems present a virtualized view of the entire system hardware to allow multiple instances of operating systems to run simultaneously on the same hardware. When that emulation is done well, the operating systems cannot distinguish simulated hardware from actual hardware. ... In some contexts a completely separate instance of the operating system is overkill for the level of isolation that is needed.” EX1058 (Hope), p. 3; *see also* §VI.C above (Technology Overview – Virtualization).

119. While chroot is limited yet simple, whole OS virtualization is fairly flexible but can be too costly for some uses. “Jails, by comparison, fit neatly in between. Jails offer a very compelling cost-benefits ratio for certain classes of problems where a fully virtualized system is too much cost or trouble but chroot is not robust enough.” EX1058 (Hope), p. 3.

120. “Jails enable system administrators to build relatively safe sandboxes that feel like virtual environments but have very low computational overhead.” EX1058 (Hope), p. 9.

121. “[T]he operating system restrictions on jails far exceed a *chroot* environment and feel more like a virtual machine. The kernel mediates access to global system information and network resources, controls the creation and use of special devices, and logically isolates jailed processes from the main system and from each other. This makes jails safer from a security point of view, and gives jails a more complete feeling of isolation.” EX1058 (Hope), p. 3.

122. “Processes in a jail are limited to a specific set of TCP/IP socket operations and a single IP address.” EX1058 (Hope), p. 4.

123. “Jails Limit File System and Device Access.” EX1058 (Hope), p. 4. “Jails Mask Processes and ID Spaces [] Jails are isolated from each other and from the host environment itself.” EX1058 (Hope), p. 4.

124. FreeBSD recognized that “[w]hile mechanisms such as *chroot* provide a modest level of compartmentalization, this mechanism has serious shortcomings, both in the scope of its functionality and effectiveness at what it provides.” EX1061 (McKusick), pp. 124-125. That is, *chroot* provided some measure of sandboxing, but it was possible to escape the sandbox’s confines. Therefore, in FreeBSD “[t]wo changes to *chroot* were made to detect and thwart these escapes.” *Id.*, p. 125.

125. Still, “[e]ven with stronger semantics, the *chroot* system call is insufficient to provide complete partitioning. ... To provide a secure virtual machine environment, FreeBSD added a new ‘jail’ facility built on top of *chroot*. Processes in a jail are provided full access to the files that they may manipulate, processes they may influence, and network services they may use. They are denied access to and visibility of files, processes, and network services outside their jail ....” *Id.*

126. McKusick (EX1061) describes the “Jail Semantics.” “Two important goals of the jail implementation are to: 1. Retain the semantics of the existing discretionary access-control mechanisms. 2. Allow each jail to have its own superuser administrator whose activities are limited to the processes, files, and network associated with its jail”. EX1061 (McKusick), p. 125.

127. Some properties of BSD Jails include the following: “Each jail is bound to a single IP address. ... A jail takes advantage of the existing *chroot* behavior to limit access to the filesystem name space for jailed processes. ... Processes within the jail will find that they are unable to interact or even verify the existence of processes outside the jail. ... Jailed processes are subject to the normal restrictions present for any processes including resource limits and limits placed by the network code, including firewall rules. ... The jail environment is a subset of the host environment. ... Processes running with superuser privileges will find that many restrictions apply to the privileged calls they may make. ...”, listing several

restrictions such as creating devices, un/mounting file systems, modifying the kernel, and more. *Id.*, pp. 126-127. *See also, generally*, EX1061 (McKusick), pp. 123-129.

## **2. Hypervisors and Virtual Machines**

128. Hypervisors, by their very nature, sandbox one virtual machine from another. *See* §VI.C above (Technology Overview – Virtualization), §VI.D above, ¶114 above (Technology Overview – Operating Systems, Sandboxing and Security for Software Implemented and Downloaded Over the Internet).

## **3. Java Virtual Machines and Java Applications**

129. In 1991, Sun Microsystems created a software language called “Java” to provide “a method to execute programs without any platform dependence” and whose “popularity soared in 1994 when it was used across the Internet.” EX1034 (Cole – Network Security Bible), 304.

130. Java’s security model provides another well-known example of a sandboxed operating system, in that the runtime environment (typically a “Java Virtual Machine (JVM)”) “resides on the host computer..., and is designed to confine [all Java applications] to a small play area... the sandbox”, which “contains no critical resources” of the host computer and for which “[a]ll access is explicitly granted by the user.” EX1034 (Cole), 304-305; *see also* EX1034 (Cole), 406 (“One of the reasons that Java is so popular is because of its intrinsic security

mechanisms... The Java Virtual Machine (JVM) is responsible for stopping buffer overflows, the use of un-initialized variables, and the use of invalid opcodes.”).

131. Garfinkel similarly explains Java’s security model. EX1045 (Garfinkel – Web Security & Commerce), 44-45 (“Java employs a variety of techniques to limit what a downloaded program can do. The main ones are the Java sandbox, the SecurityManager class, the Bytecode Verifier, and the Java Class Loader. These processes are illustrated in Figure 3-2 ... Java programs are prohibited from directly manipulating a computer’s hardware or making direct calls to the computer’s operating system. Instead, Java programs run on a virtual computer inside a restricted virtual space. Sun termed this approach to security the Java ‘sandbox,’ likening the Java execution environment to a place where a child can build things and break things and generally not get hurt and not hurt the outside world.”).

132. Tanenbaum 2001 similarly explains Java’s operation. EX1044 (Tanenbaum 2001), 642-643 (“The Java programming language and accompanying run-time system were designed to allow a program to be written and compiled once and then shipped over the Internet in binary form and run on any machine supporting Java. Security was a part of the Java design from the beginning. ... Java programs are compiled to an intermediate binary code called **JVM (Java Virtual Machine) byte code**. ... In the Java model, applets sent over the Internet for remote execution are JVM programs.”).

133. Specifically, the “Java sandbox” was well-known to be “composed of...:

- **Permissions** – Explicit statements of actions that applications are allowed to execute and resources that they are allowed to access;
- **Protection domains** – Collections of permissions that describe what actions applications from domains are allowed to execute and resources that they are allowed to access;
- **Policy files** – Contains protection domains
- **Code stores** – The sites that the applications are physically stored on prior to execution on the host
- **Certificates** – Used to sign code to convey trust to a user that you are the developer of an application
- **Key Stores** – Files that contain the certificates... Key Stores are queried to identify who signed the application code.”

EX1034 (Cole), 305.

134. Hall also discusses Java’s security sandboxing. EX1009 (Hall), 438 (“[T]he Java platform has built-in support for secure sandboxes.”).

**E. Security for Software Implemented and Downloaded over the Internet Including Digital Signature Implementations**

135. Challenged Claim 18 recites “wherein the at least one programmable network device and the programmable cloud device are adapted to verify the authenticity and integrity of updates to the plurality of network device applications and the plurality of cloud applications.” This was also a well-known security tool

to POSITAs for software implemented and downloaded over the Internet, including for Java applications hosted in a Java Virtual Machine (JVM) running on an operating system of a networked device (e.g., server).

136. As identified immediately above, Java’s “sandbox” used “certificates”, having application developer’s “sign” their applications, or a “hash” of the application (computed using standardized algorithms (e.g., SHA, MD5, etc.), using “digital signatures”, and “key stores.” *Id.*; see also EX1044 (Tanenbaum 2001), 643-644 (“Java programs are compiled to an intermediate binary code called **JVM (Java Virtual Machine) byte code** ... In the Java model, applets sent over the Internet for remote execution are JVM programs. When an applet arrives, it is run through a JVM byte code verifier that checks if the applet obeys certain rules ... **JDK (Java Development Kit)** ... 1.2 provides a configurable fine-grain security policy that applies to all applets, both local and remote. ... Each applet is characterized by two things: where it came from and who signed it. Where it came from is its URL; who signed it is which private key was used for the signature. Each user can create a security policy consisting of a list of rules. A rule may list a URL, a signer, an object, and an action that the applet may perform on the object if the applet’s URL and signer match the rule. ... To verify the signatures, [the recipient] must either have the necessary public keys on [its] disk or must acquire them dynamically, for example in the form of a certificate signed by a company [it] trusts

and whose public key [it] already has.”) (original emphasis); §VI.B.1 above (Technology Overview – Hash Functions and Their Uses).

137. As I explain in ¶¶138-147 below, these “certificate”, “digital signature” and “key store” security techniques were well-known to POSITAs, and widely implemented (including outside of Java), in order for a recipient and implementer of a downloaded application to verify the “authenticity” and “integrity” of the application, and of updates to such application (also commonly referred to as “patches”, “upgrades”, or “bug fixes”), prior to installation of the application.

138. For example, Garfinkel explains the use of digital signatures for “software that’s distributed over the Internet” (including updates (“patches”) of such software). EX1045 (Garfinkel), 169-172 (“Code signing is a technique for signing executable programs with digital signatures. Code signing is designed to improve the reliability of software that’s distributed over the Internet. It is meant to provide a system for trusting downloaded code and reducing the impact of malicious programs ... Walk into a computer store and buy a copy of Microsoft Windows 95; and you’re pretty sure that you know what you are buying and who produced it. The program, after all, comes shrink-wrapped in a box, with a difficult-to-forge security hologram. ... The same can’t be said for most software that’s downloaded over the Internet. ... Code signing is supposed to bring the assurance of shrink-wrapped software to the world of software that’s distributed electronically. It does this by

adding two things to an executable: • A digital signature that signs the executable with a secret key. • A digital certificate, which contains the corresponding public key, the name of the person or organization to whom that key belongs, and a digital signature signed by a recognized certification authority. ... Ideally, all code signatures should be verified when each piece of code is downloaded as well as before each time it is run.”), 265-266 (“Before you install any [software] patch, be sure that the patch is an authentic patch provided by your vendor ... [by] checking [its] digital signatures and cryptographic checksum to determine the authenticity and integrity of a patch before you install it.”).

139. As another example, Tanenbaum explains the use of certificates and digital signatures. EX1044 (Tanenbaum 2001), 590-591 (“One common method [to distribute public keys] is for message senders to attach a **certificate** to the message, which contains the user’s name and public key and digitally signed by a trusted third party. Once the user has acquired the public key of the trusted third party, he can accept certificates from all senders who use this trusted third party to generate their certificates.”) (original emphasis), 641-642 (“[T]here has to be a way for a user to determine that an applet was written by a trustworthy vendor and not modified by anyone after being produced. This is done using a digital signature, which allows the vendor to sign the applet in such a way that future modifications can be detected.”).

140. Indeed, the Open Standards Gateway Initiative (OSGi) even standardized the use of digitally signed software. EX1050 (“OSGi 2000 Overview”), 8-10 (“The first OSGi specifications will concentrate on Java APIs ... A bundle is a Java archive (JAR) file ... The JAR manifest file format is extended to contain the additional information about the bundle and the standard JAR signing technique is used to ensure the integrity and authenticity of downloaded bundles.”); EX1009 (Hall), 457 (“Digitally signed bundles can ... help you do two things: ■ Authenticate the provider of a bundle ■ Ensure that bundle content hasn’t been modified.”).

141. In these well-known application security schemes, an application developer would obtain two “mathematically related keys: a public key and a private key” from a trusted “certificate authority”, where the “public key is shared with others in the form of a certificate... as defined by the X.509 ITU-T standard”, on which the trusted authority certifies that the named entity in the certificate is the holder of the unique public-key-private-key-pair. EX1009 (Hall), 458.

142. Garfinkel similarly discusses code signing. EX1045 (Garfinkel), 170-172 (“Code signing is supposed to bring the assurance of shrink-wrapped software to the world of software that’s distributed electronically. It does this by adding two things to an executable: • A digital signature that signs the executable with a secret key. • A digital certificate, which contains the corresponding public key, the name

of the person or organization to whom that key belongs, and a digital signature signed by a recognized certification authority. ... These are shown in Figure 9-1.”) (which I have reproduced below):

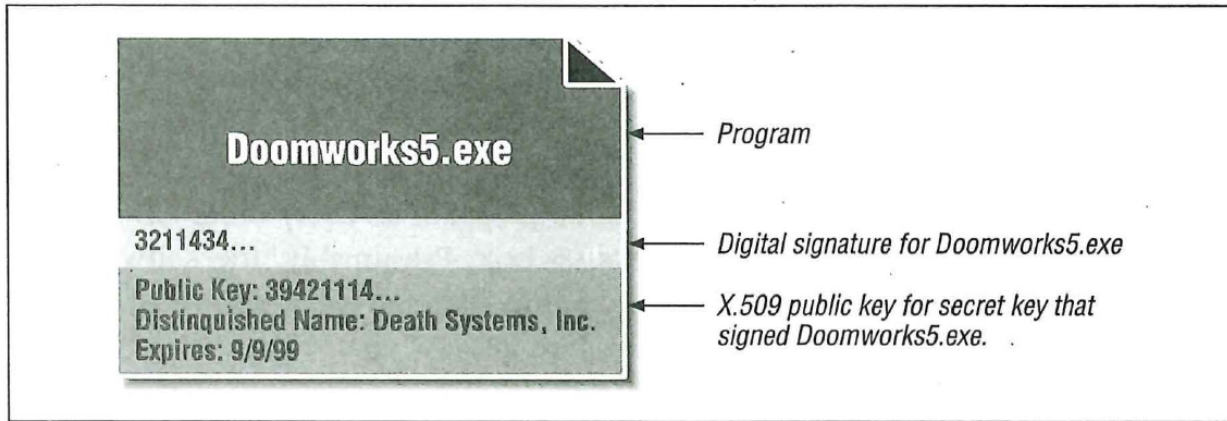
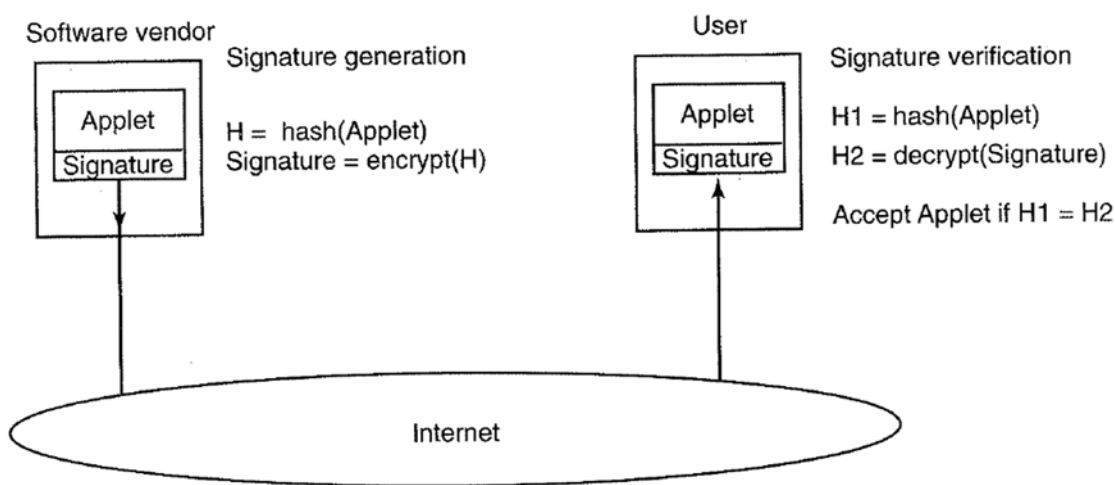


Figure 9-1. An idealized diagram of a piece of signed code, showing the code's digital signature and the corresponding digital certificate

143. Tanenbaum similarly discusses code signing. EX1044 (Tanenbaum 2001), 641-642 (“Code signing is based on public-key cryptography. An applet vendor, typically a software company, generates a (public key, private key) pair, making the former public and zealously guarding the latter.”); *see also* §VI.B.1 above (Technology Overview – Hash Functions and Their Uses).

144. In implementing this standardized security scheme for application developers and implementers, it was well known to POSITAs that the application developer would use the private key, of the unique public-key-private-key-pair issued by the trusted authority to the particular developer as certified in the associated “certificate”, to digitally sign a “one-way hash” of each of the applications that it develops, and of a “one-way-hashes” of each of the

updates/patches/upgrades to such applications, and have the recipients and implementers of such applications use the corresponding unique public key, identified on the developer's "certificate", to perform digital signature matching, prior to installation of such application (or application update), to verify that the application is authentic and unmodified:



**Figure 9-20.** How code signing works.

EX1044 (Tanenbaum 2001), Figure 9-20 (which I have reproduced above)

145. Tanenbaum details the process as follows. *Id.*, 641-642 ("Code signing is based on public-key cryptography. An applet vendor, typically a software company, generates a (public key, private key) pair, making the former public and zealously guarding the latter. To sign an applet, the vendor first computes a hash function of the applet to get a 128-bit or 160-bit number, depending on whether MD5 or SHA is used. It then signs the hash value by encrypting it with its private key

(actually, decrypting it using the notation of Fig. 9-3). This signature accompanies the applet wherever it goes. When the user gets the applet, [it] computes the hash function itself. It then decrypts the accompanying signature using the vendor's public key and compares what the vendor claims the hash function is with what ... itself computed. If they agree, the applet is accepted as genuine. Otherwise it is rejected as a forgery. The mathematics involved makes it exceedingly difficult for anyone to tamper with the applet in such a way as its hash function will match the hash function that is obtained by decrypting the genuine signature. It is equally difficult to generate a new false signature that matches without having the private key. The process of signing and verifying is illustrated in Fig. 9-20.”).

146. Tanenbaum continues to detail the process as follows. *Id.*, 590-591 (“Digital signatures make it possible to sign email messages and other digital documents in such a way that they cannot be repudiated by the sender later. One common way is to first run the document through a one-way hashing algorithm that is very hard to invert. The hashing function typically produces a fixed-length result independent of the original document size. The most popular hashing functions used are MDS (Message Digest), which produces a 16-byte result (Rivest, 1992) and SHA (Secure Hash Algorithm), which produces a 20-byte result (NIST, 1995). The next step assumes the use of public-key cryptography as described above. The document owner then applies his private key to the hash to get  $D(hash)$ . This value, called the

**signature block**, is appended to the document and sent to the receiver, as shown in Fig. 9-3. ... When the document and hash arrive, the receiver first computes the hash of the document using MD5 or SHA, as agreed upon in advance. The receiver then applies the sender's public key to the signature block to get  $E(D(hash))$ . In effect, it encrypts the decrypted hash, canceling it out and getting the hash back. If the computed hash does not match the hash from the signature block, the document, the signature block, or both have been tampered with (or changed by accident). The value of this scheme is that it applies (slow) public-key cryptography only to a relatively small piece of data, the hash. ... One common method [to distribute public keys] is for message senders to attach a **certificate** to the message, which contains the user's name and public key and digitally signed by a trusted third party. Once the user has acquired the public key of the trusted third party, he can accept certificates from all senders who use this trusted third party to generate their certificates.") (emphasis in original).

147. Garfinkel describes similar processes. EX1045 (Garfinkel), 170-172 (“[C]ode signing presupposes the existence of a working public key infrastructure. Otherwise, there is no way to tell whose signature is on a piece of signed code. To be useful, the signatures must be verified. ... The need for code signing was demonstrated in 1995 and 1996, when a program called *PKZIP30B.EXE* was reportedly uploaded to many software libraries on the Internet. The program

appeared to be the 3.0 Beta release of PKZIP, a popular disk compression program. But when unsuspecting users downloaded the program and tried to run it, the rogue program actually erased the user's hard disk. That didn't exactly do wonders for the reputation of PKWare, PKZIP's creator. Digital signatures could have prevented this mishap. If PKWare had previously adopted a policy of signing their programs with the company's digital signature, then bad hackers could never have created the *PKZIP30B.EXE* file and credibly passed it off as being the creation of PKWare. That's because the rogue hackers who distributed the *PKZIP30B.EXE* program couldn't have signed it with PKWare's digital signature. ... Today [1997] ... Companies can simply obtain a software publisher's certificate from a recognized certification authority and use it to sign the programs that they distribute.”), 265-266 (“Before you install any patch, be sure that the patch is an authentic patch provided by your vendor. You can often check a file's digital signatures and cryptographic checksum to determine the authenticity and integrity of a patch before you install it.”); *see also* §VIII.A.3 below (Overview of the Prior Art in the Grounds – The Hall Textbook (EX1009)).

#### **F. Distributed Systems**

148. The term “distributed system” in the context of computer networks has long been known to POSITAs as “a software system built on top of a network”, and where “the distinction between a network and a distributed system lies with the

software (especially the operating system), rather than the hardware.” Moreover, the “user of a **distributed system** is not aware that there are multiple processors, it looks like a virtual uniprocessor.” EX1032 (Tanenbaum 1996), 2.

149. Software “modularity”—where “the code of [a single] software application is divided into logical parts representing separate concerns”—was long understood (since at least the 1970s) as an important concept for building a software system on top of a network. EX1009 (Hall), 4-5. Indeed, it was well known to POSITAs that modular software solutions “provid[e] isolation in a multi-tenancy context”, “make[] developing software more scalable”, and “enable[] structuring of development teams such that parallel development of modules, which together form the application, is achievable.” EX1008 (Alves), xiii. Moreover, because “[a] module’s internals are inaccessible to the outside,... tends to lead to well-defined APIs that better allow for concurrent development.” *Id.*

150. Sosinsky similarly highlights the importance of software “modularity” to distributed **software systems** (including those implemented using **cloud servers**). EX1052 (Sosinsky), 46-47 (“Applications built in the cloud often have the property of being built from a collection of components, a feature referred to as composability. A composable system uses components to assemble services that can be tailored for a specific purpose using standard parts. A composable component must be: • **Modular:** It is a self-contained and independent unit that is cooperative,

reusable, and replaceable. ... Although cloud computing doesn't require that hardware and software be composable, it is a highly desirable characteristic from a developer or user's standpoint, because it makes system design easier to implement and solutions more portable and interoperable. ... A PaaS or SaaS service provider gets the same benefits from a composable system that a user does- these things, among others: • Easier to assemble systems; • Cheaper system development; • More reliable operation; • A larger pool of qualified developers; • A logical design methodology. You encounter the trend toward designing composable systems in cloud computing in the widespread adoption of what has come to be called the Service Oriented Architecture (SOA). The essence of a service oriented design is that services are constructed from a set of modules using standard communications and service interfaces.”), 271 (“[A]s cloud computing applications expand their capability to provide additional and diverse services, SOA offers access to ready-made, modular, highly optimized, and widely shareable components that can minimize developer and infrastructure costs.”).

151. Clouds are but the latest form of “distributed computing” systems or “distributed processing.” EX1057 (Microsoft Computed Dictionary), p. 153.

152. Generally, a “distributed network” is “[a] network in which processing, storage, and other functions are handled by separate units (nodes) rather than by a single main computer.” EX1057 (Microsoft Computed Dictionary), p. 154.

153. The “Distributed Computing Environment” is “[a] set of standards from the Open Group (formerly the Open Software Foundation) for development of distributed applications that can operate on more than one platform. *Acronym*: DCE. *See also* distributed processing.” EX1057 (Microsoft Computed Dictionary), pp. 153-154.

154. “[D]istributed processing” is “[a] form of information processing in which work is performed by separate computers linked through a communications network. Distributed processing is usually categorized as either plain distributed processing or true distributed processing. Plain distributed processing shares the workload among computers that can communicate with one another. True distributed processing has separate computers perform different tasks in such a way that their combined work can contribute to a larger goal. The latter type of processing requires a highly structured environment that allows hardware and software to communicate, share resources, and exchange information freely.” EX1057 (Microsoft Computed Dictionary), p. 154.

155. The evolution of distributed systems has been taught to computer scientists for years (well before they even became a POSITA as I have defined such a person in §V above (“Person of Ordinary Skill in the Art”).

156. As soon as computers could communicate with each other, networks of computers began to evolve incrementally over the years—from simple client/server

configurations, to more complex clusters of computers, to managed private/public clusters nowadays called clouds. *See, generally*, EX1063 (Stallings), pp. 677-712.

157. Sosinsky provides more details on clouds. EX1052 (Sosinsky), xxv (“In reality, the cloud is something that you have been using for a long time now; it is the Internet, along with all the associated standards and protocols that provide a set of Web services to you. When you draw the Internet as a cloud, you are representing one of the essential characteristics of cloud computing: abstraction. In the cloud, resources are pooled and partitioned as needed, and communications are standards-based.”).

158. Sosinsky begins to explain the history of cloud’s evolution. *Id.*, 19 (“When you consider the development of cloud computing to date, it is clear that the technology is the result of the convergence of many different standards. ... The cloud computing industry is working with these architectural standards: • Platform virtualization of resources; • Service-oriented architecture; • Web-application frameworks; • Deployment of open-source software; • Standardized Web services; • Autonomic systems; • Grid computing.”).

159. Sosinsky continues to explain how many technologies, protocols, and standards came together over several decades to form cloud computing. *Id.*, 45 (“Cloud computing is a natural extension of many of the design principles, protocols, plumbing, and systems that have been developed over the past 20 years. However,

cloud computing describes some new capabilities that are architected into an application stack and are responsible for the programmability, scalability, and virtualization of resources. One property that differentiates cloud computing is referred to as composability, which is the ability to build applications from component parts. A platform is a cloud computing service that is both hardware and software. Platforms are used to create more complex software. Virtual appliances are an important example of a platform, and they are becoming a very important standard cloud computing deployment object. Cloud computing requires some standard protocols with which different layers of hardware, software, and clients can communicate with one another. Many of these protocols are standard Internet protocols. Cloud computing relies on a set of protocols needed to manage interprocess communications that have been developed over the years. The most commonly used set of protocols uses XML as the messaging format, the Simple Object Access Protocol (SOAP) protocol as the object model, and a set of discovery and description protocols based on the Web Services Description Language (WSDL) to manage transactions.”).

160. Sosinsky then explains how cloud computing is an orthogonal, complementary solution to service-oriented architectures. *Id.*, 272 (“Cloud computing is not the next evolutionary step beyond SOA, as some think because of SOA’s longer history. The two technologies are complementary. Whereas SOA can

be used to construct large and complex applications that scale both horizontally and vertically, cloud computing applications tend to be scaled vertically. Horizontal scaling refers to applications with a large number of different business processes operating. Vertical scaling refers to large applications with a limited number of business processes operating. SOA techniques may be applied in both instances

161. “[A] *client/server environment* is populated by clients and servers. The **client** machines are generally single-user PCs or workstations that provide a highly user-friendly interface to the end user. ... Each **server** in the client/server environment provides a set of shared services to the clients.” EX1063 (Stallings), p. 678. As I discussed in §VI.A.1 above, an end-point computer is an example of a client, but note that a computer can be both a client and a server (*e.g.*, a proxy device), and that a server could be referred to as a client (*e.g.*, in **distributed systems**) depending on its role.

162. “In addition to clients and servers, the third essential ingredient of the client/server environment is the **network**. Client/server computing is typically distributed computing. Users, applications, and resources are distributed in response to business requirements and linked by a single LAN or WAN or by an internet of networks.” EX1063 (Stallings), p. 679.

163. “The traditional client/server architecture involves two levels, or tiers: a client tier and a server tier. A three-tier architecture is also common (Figure 16.6).

In this architecture, the application software is distributed among three types of machines: a user machine, a middle-tier server, and a backend server.” EX1063 (Stallings), p. 685.

164. The evolution of networked systems continued. “The service-oriented architecture (SOA) is a form of client/server architecture that now enjoys widespread use in enterprise systems. An SOA organizes business functions into a modular structure rather than as monolithic applications for each department. As a result, common functions can be used by different departments internally and by external business partners as well. The more fine-grained the modules, the more they can be reused. In general, an SOA consists of a set of services and a set of client applications that use these services. A client request may involve a single service or may involve two or more services to coordinating some activity, requiring communication of services with each other. The services are available through published and discoverable interfaces.” EX1063 (Stallings), p. 689.

165. Sosinsky similarly explains service-oriented architectures. EX1052 (Sosinsky), 47 (“You encounter the trend toward designing composable systems in cloud computing in the widespread adoption of what has come to be called the Service Oriented Architecture (SOA). The essence of a service oriented design is that services are constructed from a set of modules using standard communications and service interfaces. An example of a set of widely used standards describes the

services themselves in terms of the Web Services Description Language (WSDL), data exchange between services using some form of XML, and the communications between the services using the SOAP protocol.”).

166. Sosinsky then details SOAs. *Id.*, 271 (“Service Oriented Architecture (SOA) describes a standard method for requesting services from distributed components and managing the results. Because the clients requesting services, the components providing the services, the protocols used to deliver messages, and the responses can vary widely, SOA provides the translation and management layer in an architecture that removes the barrier for a client obtaining desired services. With SOA, clients and components can be written in different languages and can use multiple messaging protocols and networking protocols to communicate with one another. SOA provides the standards that transport the messages and makes the infrastructure to support it possible. SOA provides access to reusable Web services over a TCP/IP network, which makes this an important topic to cloud computing going forward. ... Service Oriented Architecture (SOA) is a specification and a methodology for providing platform and language-independent services for use in distributed applications. A service is a repeatable task within a business process, and a business task is a composition of services. SOA describes a message-passing taxonomy for a component-based architecture that provides services to clients upon demand. ... Usually service providers and service consumers do not pass messages

directly to each other. Implementations of SOA employ middleware software to play the role of transaction manager (or broker) and translator. That middleware can discover and list available services, as well as potential service consumers, often in the form of a registry, because SOA describes a distributed architecture security and trust services are built directly into many of these products to protect communication. Middleware products also can be where the logic of business processes reside; they can be general-purpose applications, industry-specific, private, or public services.”).

167. Next came “Distributed Message Passing” using “two most common approaches. The first is the straightforward application of messages as they are used in a single system. The second is a separate technique that relies on message passing as a basic function: the remote procedure call.” EX1063 (Stallings), p. 691.

168. “A variation on the basic message-passing model is the remote procedure call. This is now a widely accepted and common method for encapsulating communication in a distributed system. The essence of the technique is to allow programs on different machines to interact using simple procedure call/return semantics, just as if the two programs were on the same machine. That is, the procedure call is used for access to remote services.” EX1063 (Stallings), p. 695.

169. The next step in this evolution were the clusters. “Clustering is an

alternative to symmetric multiprocessing (SMP) as an approach to providing high performance and high availability and is particularly attractive for server applications. We can define a cluster as a group of interconnected, whole computers working together as a unified computing resource that can create the illusion of being one machine. The term *whole computer* means a system that can run on its own, apart from the cluster; in the literature, each computer in a cluster is typically referred to as a *node*.” EX1063 (Stallings), p. 699.

170. Clusters’ benefits were fairly similar to those of clouds. Stallings, citing to a 1997 paper by E. Brewer, “lists four benefits that can be achieved with clustering. ... • **Absolute Scalability** ... • **Incremental Scalability** ... • **High availability** ... • **Superior price/performance.**” EX1063 (Stallings), p. 699 (internal citation omitted).

171. Clusters can be configured in many ways and also support failure management and load balancing. *See, generally*, EX1063 (Stallings), pp. 699-704.

172. Several examples of clusters include the following. The Microsoft “Windows Failover Clustering is a shared-nothing cluster, in which each disk volume and other resources are owned by a single system at a time.” EX1063 (Stallings), p. 704.

173. Here, “[a] group combines resources into larger units that are easily managed, both for failover and load balancing. Operations performed on a group,

such as transferring the group to another node, automatically affect all of the resources in that group. Resources are implemented as dynamically linked libraries (DLLs) and managed by a resource monitor. The resource monitor interacts with the cluster service via remote procedure calls and responds to cluster service commands to configure and move resource groups. Figure 16.17 depicts the Windows clustering components and their relationships in a single system of a cluster. The **node manager** is responsible for maintaining this node's membership in the cluster." EX1063 (Stallings), p. 705.

174. Another example is the Beowulf project and its Linux clusters. "In 1994, the Beowulf project was initiated under the sponsorship of the NASA High Performance Computing and Communications (HPCC) project. Its goal was to investigate the potential of clustered PCs for performing important computation tasks beyond the capabilities of contemporary workstations at minimum cost. Today, the Beowulf approach is widely implemented and is perhaps the most important cluster technology available. ... Key features of Beowulf include the following [RIDG97]: • Mass market commodity components • Dedicated processors (rather than scavenging cycles from idle workstations) • A dedicated, private network (LAN or WAN or internetted combination) • No custom components • Easy replication from multiple vendors • Scalable I/O • A freely available software base • Use of freely available distribution computing tools with minimal changes

- Return of the design and improvements to the community.” EX1063 (Stallings), pp. 706-707.

175. Beowulf Linux clusters supported distributed applications. “**Beowulf distributed process space (BPROC)**: This package allows a process ID space to span multiple nodes in a cluster environment and also provides mechanisms for starting processes on other nodes. The goal of this package is to provide key elements needed for a single system image on Beowulf cluster. BPROC provides a mechanism to start processes on remote nodes without ever logging into another node and by making all the remote processes visible in the process table of the cluster’s front-end node.” EX1063 (Stallings), p. 708.

176. As **distributed systems** grew in popularity and importance, like many other efforts on the Internet, a group was formed to standardize this, described next.

### 1. Open Standards Gateway Initiative (OSGi)

177. A well-known initiative related to modular software and **distributed systems**—the Open Standards Gateway Initiative (OSGi)—began as a three-year joint research project in the late 1990s between Swedish companies Ericsson and CR&T, from which, in 1999, (1) a patent application describing foundational concepts was filed (EX1015/EX1016/EX1004 (collectively – “Vasell”, *see* §VIII.A.1 below (Overview of the Prior Art in the Grounds –Vasell)), (2) the assignee of this patent application (Gatespace AB) was founded, and (3) the non-

profit OSGi alliance was founded.

178. For example, EX1017 (About Gatespace 2000) states that “The founders of Gatespace have been active in the technical work of OSGi since [it] started in May 1998, and Gatespace is a charter member of OSGi. Gatespace was formed by CR&T... in July 1999 as an independent software company.” EX1017 (About Gatespace 2000), at 1.

179. A provisional application was filed June 8, 1998, which named founders of Gatespace, and other Ericsson and CR&T employees, as inventors. *See* EX1015 (Vasell 1<sup>st</sup> Provisional), 1-2 (the named inventors are Malte Lilliestråle, Hans Thorsen, Jesper Vasell, Staffan Truvé, Carlo Pompilii, Johan Ljungberg, Jorgen Andersson).

180. Additional details on the assignees and inventors are available and establish the close relationships between the Vasell patent filings, Gatespace, and OSGi. *See* §VIII.A.1 below (Overview of the Prior Art in the Grounds – Vasell); EX1016 (Vasell 2<sup>nd</sup> Provisional), 1-3 (filed March 12, 1999; named inventors and authors are Malte Lilliestråle, Tom Idermark, and Jesper Vasell; includes article titled “Ericsson’s e-box system-An electronic services enabler”); EX1004 (Vasell Patent), 1 (filed June 7, 1999; named Assignee is “Gatespace AB”; named inventors are Jesper Vasell, Tom Idermark, Malte Lilliestråle, Hans Thorsen, Staffan Truvé, Carlo Pompilii, Johan Ljungberg, Jorgen Andersson; claims priority to, and

incorporates by reference in their entirety, EX1015 (Vasell 1<sup>st</sup> Provisional) and EX1016 (Vasell 2<sup>nd</sup> Provisional)).

181. Gamespace's formation and purpose are described as follows (and show its close ties to OSGi). EX1018 (Gamespace Formation Press Release), 1 ("Press Release 1999-09-30 ... Carlstedt Research & Technology AB (CR&T); has formed Gamespace AB, a new company targeted towards developing software for the 'intelligent home' and other Internet based service systems in cooperation with Ericsson Radio Systems AB. The new company will develop products based on the new proposed standard for the area of networked based services, 'Open Service Gateway Initiative', which is being proposed by Ericsson and other industry leaders. The company will develop this standard further, and develop software platforms, development tools, and management systems. Target customers include providers of networked services, network operators, and third party developers.").

182. Gamespace described several key management personnel (which were key management personnel from Ericsson and CR&T and named inventors on the Vasell patent filings). EX1021 (CR&T People 2000), 1 (Staffan Truvé and Jesper Vasell, named Vasell inventors, were "Managing Director" and "Vice Managing Director" of CR&T respectively); EX1024 (OSGi Leadership 2001), 1 ("May 23, 2001 ... the Open Services Gateway Initiative (OSGi) elected a new board of directors and officers to serve the organization through May 2002" including

“Staffan Truvé, CEO, Gatespace” (named Vasell inventor (EX1015, Vasell 1<sup>st</sup> Provisional; EX1004, Vasell Patent)); EX1025 (Gatespace Management 2002), 1 (“Staffan Truvé is co-founder, CEO and President for Gatespace” and “is currently Vice President of Marketing and a board member of OSGi. In 1994 he co-founded Carlstedt Research and Technology.”).

183. OSGi’s officers included key management personnel from Gatespace and Ericsson (who were named inventors on the Vasell patent filings). EX1023 (OSGi Officers 2000), 1 (OSGi “Officers (June 1999 - June 2000)” included “Regional Vice President Europe, Middle East and Africa: Mr. Malte Lilliestråle, Ericsson” (a named Vasell inventor (*see, generally* EX1015 (Vasell 1<sup>st</sup> Provisional), EX1016 (Vasell 2<sup>nd</sup> Provisional), EX1004 (Vasell Patent))).

184. Gatespace’s press release provides quotes from its leadership regarding its close ties to Ericsson and OSGi. EX1018 (Gatespace Formation Press Release), 1 (“Staffan Truvé, managing director CR&T; We see this as a natural extension of a long term collaboration, where CR&T; has developed the software architecture for Ericsson’s e-box product. ... Malte Lilliestråle, Ericsson Radio Systems AB, business development e-box. ... The new company will offer products that enable rapid development and deployment of new services for e-box and other OSGI compliant systems.”).

185. Ericsson and OSGi further explain their relationship. EX1019

(Ericsson E-Services WebPage), 1 (“Ericsson e-services is a part of the Open Services Gateway Initiative (OSGI).”); EX1020 (OSGi Announcement on Ericsson E-Services WebPage), 1 (“March 1, 1999. Fifteen leading technology companies today announced a new alliance to create and maintain the Open Service Gateway specification”, which “will be designed to provide a common foundation for Internet service providers (ISPs), network operators and equipment manufacturers to deliver a wide range of Internet services to gateway servers running in the home or remote office. ... By writing to the Open Service Gateway specification’s Java technology-based environment, service providers and vendors traditionally faced with inflexible dedicated systems development will be able to leverage the infrastructure of the Internet while drawing from the resources of the millions of Java software and Internet developers worldwide.”).

186. The OSGi charter further explains who is involved, including Ericsson and the company that developed Java (Sun Microsystems). EX1022 (OSGi Charter), 1 (“The OSG Initiative (OSGi) is an open industry effort announced in March, 1999 by”, among others, “Ericsson” and “Sun Microsystems”, and with “[a]dditional companies”, including “Gatespace AB”, “hav[ing] also announced their support for OSGi during the last few months.”).

187. Overall, Gatespace describes its formation and purpose as follows (which confirms the close ties between it, Ericsson, CR&T, the Vasell patent filings

and OSGi). EX1026 (About Gatespace 2002), 1 (“Founded in 1999, Gatespace is a result of a three-year joint research project between Ericsson and the Swedish high-end research and consulting firm CR&T; (Carlstedt Research & Technology) - a project which also resulted in Ericsson’s launch of the e-box in 1999. Gatespace products are designed to be fully compliant with the OSGi (Open Services Gateway initiative) specification. We support full interoperability between content/application service providers, network operators, and service gateway and device manufacturers. This provides for secure and reliable end-to-end services. Gatespace is a member of the OSGi and was selected to provide core portions of the reference implementation of the OSGi’s first specification. When OSGi released OSGi specification 1.0 in May 2000 [*see* EX1050], Gatespace was one of the first companies to launch a product designed to comply with the OSGi specification.”).

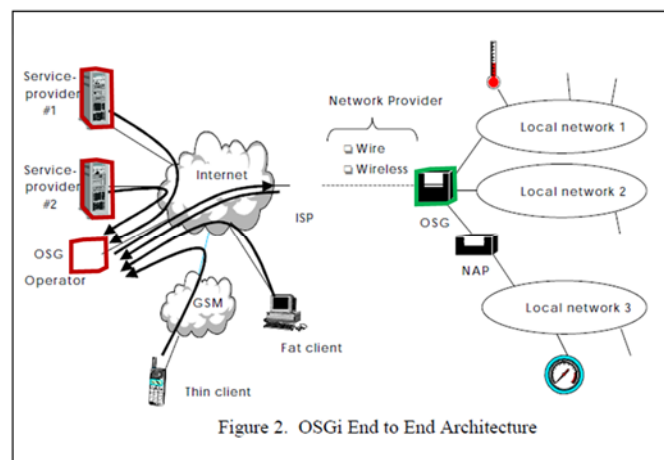
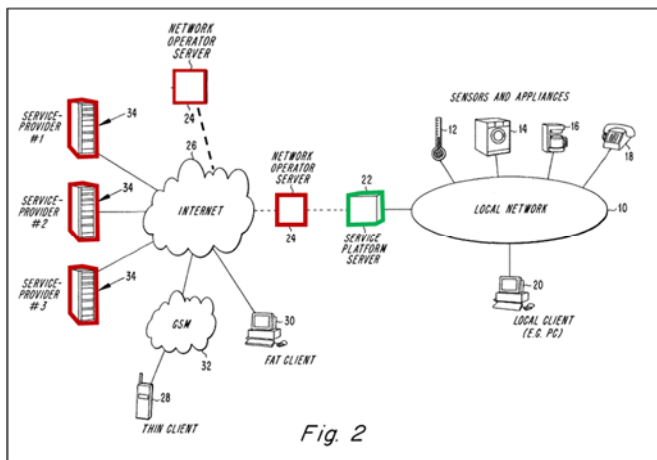
188. In 2007, “Gatespace AB” was renamed “Makewave AB” and again described the company history (which confirms the close ties between it, Ericsson, CR&T, the Vasell patent filings and OSGi). EX1027 (About Makewave 2007), 1 (“Established in ... 1999 as a joint venture between Ericsson and CR&T, Gatespace AB ... has been a member of the OSGi Alliance from instigate in 1999 and a major contributor to the various framework revisions ever since. As a founding father and owner, Ericsson was enthusiastic to use Gatespace’s platform and eventually the Swedish giant developed a full-scale system, the e-Box, based upon the OSGi

distribution. The other instigator of the joint venture, CR&T, held a high reputation as the ambassador of Swedish innovation companies ... The choice of CR&T; founder and leader, *Staffan Truvé*, to leave his position at CR&T; in order to become CEO of Gatespace in 2001 is a clear indication of how promising OSGi is as a technology ... In May 2007 the company-name was changed to Makewave AB, a name that satisfyingly showed our nature as a creator and provider of products and services that accelerates businesses world-wide.”); *see also* §§VIII.A.1-VIII.A.4 below (Overview of the Prior Art in the Grounds); §IX.A below (Motivation to Combine the Teachings of the Prior Art in the Grounds).

189. After the 1999 filing of Vasell (*see* §VIII.A.1 below), and over the succeeding decade-plus, the OSGi members, and POSITAs in academia and industry, developed and published technical implementation details for foundational concepts disclosed in Vasell in the OSGi’s open specifications (*e.g.*, EX1028 (OSGi Service Platform Enterprise Specification)) and numerous OSGi-related textbooks (*e.g.*, EX1009 (Hall, *see* §VIII.A.3 below), EX1008 (Alves, *see* §VIII.A.2 below)), whitepapers (*e.g.*, EX1013 (OSGi 2005 Whitepaper), EX1050 (OSGi 2000 Overview)), and articles (*e.g.*, EX1029 (Forst), EX1012 (Kachele, §VI.F.2 below), EX1010 (Kim, §VI.F.2 below), EX1011 (Rellermeyer, *see* §VIII.A.4 below)). *Id.*

190. The exemplary depiction of the “end-to-end architecture” in the 1999 patent application (EX1004 (Vasell Patent), FIG. 2 (below left, annotated)), and in

the first OSGi specification in January 2000 (EX1050 (OSGi 2000 Overview), Figure 2 (below right, annotated)) provides additional evidence of the foundational nature of this 1999 patent application (Vasell, *see* §VIII.A.1 below) to the OSGi specifications:

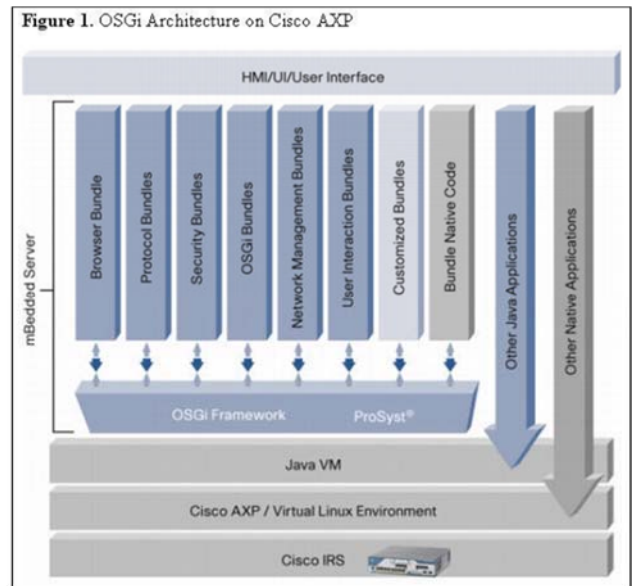
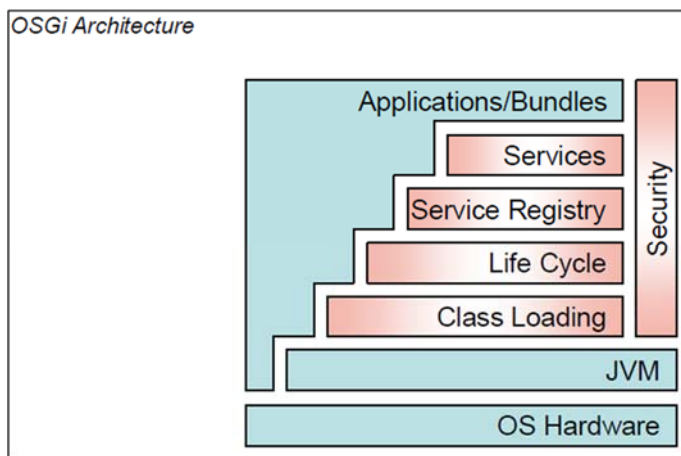


EX1004 (Vasell Patent), FIG. 2; EX1050 (OSGi 2000 Overview), FIG. 2.

191. The standardized “OSGi Service Platform” was known to POSITAs as “a dynamic module system for Java” that “allows Java code to be modularized and to be managed as services”, and the “OSGi framework” of such platform was known as “providing a lightweight and scalable solution.” EX1008 (Alves), 2, 6-7, 16. The “OSGi framework” of the standardized “OSGi Service Platform” included, in part, “a portable and secure execution environment based on” the “Java Runtime Environment (JRE), which includes the Java virtual machine that isolates the [software component] developer from the details of the underlying OS and hardware”, and “[a] dynamic module system, which can be used to dynamically

install and uninstall Java modules, which OSGi calls bundles.” EX1008 (Alves), 2, 6-7, 16.

192. Accordingly, and as I show in the example disclosures below, it was known to POSITAs that the standardized OSGi architecture was virtualized on a device (e.g., router, gateway, server, etc.), with the “OSGi framework” executing in at least one Java virtual machine (JVM) (see also §VI.D.3 above (Java Virtual Machines and Java Applications))) running on a host operating system (e.g., Linux, UNIX, etc.) OS of the device:



EX1013 (OSGi 2005 Whitepaper), 11; *id.*, 3-4; EX1029, 1-2 (Forst); EX1010 (Kim), Figure 2; see also §VI.C above (Technology Overview – Virtualization).

193. The Alves textbook (EX1008) described OSGi as a “standards-based solution to modularity” and identified it as enabling the achievement of the well-known benefits of modular software solutions, including “providing isolation in a

multi-tenancy context”, “mak[ing] developing software more scalable as modules tend to be highly focused with a clear demarcation of responsibility”, having “well-defined APIs that better allow for concurrent development”, “enabl[ing] structuring of development teams such that parallel development of modules, which together form the application, is achievable”, etc. EX1008 (Alves), xiii-xiv.

194. “Given that [OSGi] has been around since the late 1990s”, the Alves textbook (EX1008) also described that the “OSGi specifications provide[d] a very mature, stable, and comprehensive modularity solution”, which “enable[d] a dynamic system where *bundles*, the OSGi name for [software] modules, are... properly encapsulated, and loosely coupled.” EX1008 (Alves), xiv; *see also* EX1013 (OSGi 2005 Whitepaper), 13 (“A bundle is typically stored in a Java Archive, also called a JAR. Every JAR contains a Manifest” that “stores information about the contents of the JAR in headers.”), 7 (“The OSGi specifications provide the Bundle format. Bundles are applications packaged in a standard Java Archive (JAR) file, which format is compatible with the ubiquitous ZIP files.”).

195. Because the modularized OSGi framework was executed in the virtualized, hosted environment of a JVM running on the OS of a networked device (e.g., server) (*see* ¶192 above), a whitepaper published by the OSGi Alliance in 2005 (EX1013, OSGi 2005 Whitepaper) described the framework as providing various benefits, including the following four (4) benefits: (1) an “open”, “standard...

software component framework for... service providers, and developers”; (2) “[a] powerful model for... [r]unning multiple applications in the same JVM” and with “multi-vendor interoperability” in the “underlying... operating system” and device hardware; (3) “[a] secure environment that executes applications in a sandbox so that these applications cannot harm the environment, nor interfere with other resident applications”; and (4) “continuous computing” through “[a] flexible deployment Application Programming Interface (API) that controls the lifecycle of applications” whereby “[a]pplications are installed[,]. . . started, stopped, updated, and uninstalled without requiring the JVM to be restarted.” EX1013 (OSGi 2005 Whitepaper), 4, 8, 10-11.

196. In §VIII.A.3 below (Overview of the Prior art in the Grounds – The Hall Textbook), I discuss some technical implementation details that the Hall textbook (EX1009) disclosed regarding OSGi’s Enterprise specification (EX1028). Specifically, Hall (EX1009) is an example of a publication that disclosed technical implementation details for having the OSGi management software implement well-known and standardized digital signature, certificate and key exchange techniques to verify the respective authenticity and integrity of each downloaded software bundle (and of each updated bundle version).

197. Related to the “application management portal” recited in the Challenged Claims, the OSGi 2005 Whitepaper (EX1013) described the “OSGi

remote management architecture” as including, for example, a (i) “management API” and “management... bundle[s]” (“management agent”), which were “flexible” in that it could be mapped to any management protocol and permitted the “remote management” of “thousands or even millions of Service Platforms from a single management domain”, through which various management services (“lifecycle management”, “permissions”, “dynamic updates”, etc.), were provided, and (ii) “service registry” that “dynamically links different bundles together while tracking their state and dependencies.” EX1013 (OSGi 2005 Whitepaper), 4, 7-8, 10, 13-17.

198. Regarding the management API, the OSGi 2005 Whitepaper (EX1013) stated “[i]nstalling a new bundle in a remote JVM (and uninstalling it) provides the basis for networked services”, and described implementation details for these (and other) “lifecycle management” services at a high level. EX1013 (OSGi 2005 Whitepaper), 7, 13, 15-17.

199. Specifically, regarding the “[u]pdate a bundle” service, the OSGi 2005 Whitepaper (EX1013) stated that the OSGi framework “brings standardized **hot update** technology to small embedded devices... and servers enabling these devices to run continuously” and, more particularly, “first stops existing applications, after which their resources are cleaned up. Code is unloaded and replaced with the updated code. After the code is updated, the bundle is restarted. This all happens without restarting the JVM.” EX1013 (OSGi 2005 Whitepaper), 7, 10; *see also*

EX1013 (OSGi 2005 Whitepaper), 3-4, 11, 17-18.

200. Related to the “**infrastructure application marketplace**” recited in the Challenged Claims, from early on, OSGi implementers (EX1051 (2004)) and the OSGi Alliance (EX1041 (2006)) respectively created and utilized a “**bundle repository**”, as a “**central clearing house** for bundles” and to “promote a **bundle market**”. EX1051 (OBR 2004), 1; EX1041 (Kriens OSGi Blog), 1.

201. An OSGi “**bundle repository**”, contained OSGi bundles developed by the OSGi Alliance as well as third party developers, and, for OSGi framework administrators, provided a “**web based interface**” through which the administrator could (i) “search for bundles”; (ii) obtain “the description”, “more detailed information” and “comments” by “hovering over” and “clicking on” each bundle’s “link”; (iii) “select a number of bundles and *resolve* these bundles... meaning that their dependencies are used to find other bundles that can resolve those dependencies” (*i.e.*, to implement “applications built using a service oriented architecture” and requiring multiple bundles), and (iv) and “download and install” the selected and resolved bundles. EX1041 (Kriens OSGi Blog), 1; *see also* EX1051 (OBR 2004), 1.

202. An OSGi 2005 whitepaper describes the software development life-cycle. EX1013 (OSGi 2005 Whitepaper), 7-8 (describing “life-cycle management of applications and components”, and the “flexible OSGi remote management

model”, provided by the standardized OSGi framework including “install” and “update” functions), 14-15 (describing the “in-memory”, “OSGi Service Registry” provided by the standardized OSGi framework, which “enables the OSGi Service Platform to support applications built using a service-oriented architecture (SOA)”, in which services are implemented using “loosely coupled [software] components” (*i.e.*, bundles), and where the standardized “Service Registry is the glue that binds these components seamlessly together.”); *see also* §VI.F.2 below (Distributed Systems – Clouds) (*citing* EX1010 (Kim), 206, 208-209, 212, Figures 1-2); §VIII.A.2 below (Overview of the Prior Art in the Grounds – The Alves Textbook) (*citing* EX1008 (Alves), 263, 266-267).

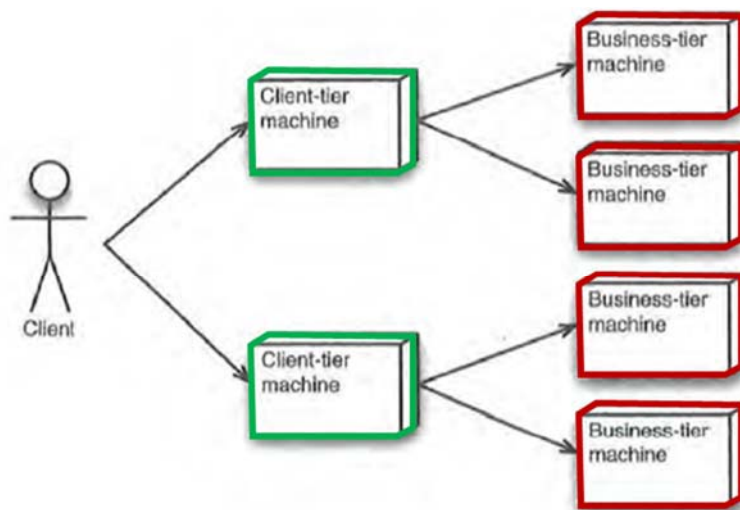
203. Commercial suppliers of OSGi technology offered software packages including, for example, (i) “certified compliant OSGi” software for running the virtualized OSGi framework on any devices; (ii) a “User Interface Package” for developers to easily “build[] modular and dynamic OSGi bundles”, and (iii) an “Application Manager” for **remotely** “**managing** OSGi-based applications” and “serving as an **Appstore**.” EX1042 (Knoplerfish Data Sheets), 1 (“Knoplerfish Pro Enterprise is certified to be OSGi Release 4, version 4.2 compliant”), 2; *see also* EX1028 (OSGi Service Platform Enterprise Specification, Release 4, version 4.2).

204. While “OSGi was initially employed in the embedded market”, POSITAs understood that, “with its growing popularity and maturity, OSGi...

created the OSGi Service Platform Enterprise Specification (Enterprise OSGi)” that “define[d] a collection of OSGi services that can be used together for enterprise features” including “management,... monitoring, and **distribution.**” EX1008 (Alves), 3-4, 8, 16; EX1028 (OSGi Service Platform Enterprise Specification); EX1013 (OSGi 2005 Whitepaper), 17-18 (2005: “The adoption of the OSGi specifications in different industries is creating a demand for new services (vertical)... The OSGi Service Platform is an excellent platform for Web Services. Its superb dynamic update facilities, the rich software environment that Java offers, and cooperative facilities that the OSGi Service Platform offers are an ideal combination with Web Services.”); *see also* §VIII.A.2 below (Overview of the Prior Art in the Grounds – The Alves Textbook) (*citing* EX1008 (Alves), xiv, 249, 252-255, 261-262, Figure 10.9).

205. As a “modular” software solution operating in a virtualized, hosted environment of a JVM running on the OS of a networked device (*e.g.*, router, gateway, server) (*see* ¶192 above), with the OSGi Service Platform Enterprise Specification (EX1028), OSGi “**easily** move[d] from a local environment... to a two-tier multiple machine environment”, whereby processing for a given service was “partition[ed]... into”, for example, “the **web application bundle** in the OSGi framework instances hosted in the **client tier machines**” and “the **auction application bundles** [hosted] in the **business tier machines**, as shown in Figure 10.10” (which I

reproduced and annotated below), and (ii) rendered “[t]he **system**... more scalable, because... we have a flexible setup, where we understand which **components** can be replicated and how best to partition the requests”:



**Figure 10.10**  
Two-tier architecture  
for the auction system

EX1008 (Alves), 261-263; *see also* §VIII.A.2 below (Overview of the Prior Art in the Grounds – Alves).

206. Related to the claimed “distributed **application**” with components installed on each “**device**” recited in the Challenged Claims, OSGi’s standardized Enterprise implementation (EX1028, OSGi Service Platform Enterprise Specification) included communications between a **locally-installed** “**bundle** that’s exporting a remote service in one OSGi framework instance and [a **remotely-installed**] **bundle** that’s importing the service in another [OSGi] framework instance” using well-known protocols “such as RMI (Remote Method Invocation), SOAP (Simple Object Access Protocol), and REST (Representational State Transfer)”.

which POSITAs understood to provide secure communications between software components of a distributed application:

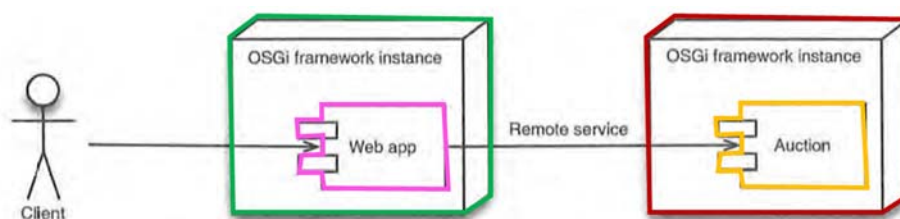


Figure 10.9 An auction system partitioned into two separate OSGi framework instances, communicating through an OSGi remote service

EX1008 (Alves), 249, 252-255, 261-262, Figure 10.9 (annotated above).

207. Figure 10.9 above shows an example of an “auction system” distributed application in which a “web app” bundle installed at a local machine securely communicates with an “auction” bundle installed at a “business tier” machine); see also §VIII.A.2 below (Overview of the Prior Art in the Grounds – Alves).

208. Well-known SOAP security is described as follows. EX1046 (Oasis - SOAP Security Specification), 1-2, 7-8, 13-15 (Because “[p]rotecting the message content from being disclosed (confidentiality) or modified without detection (integrity) are primary security concerns”, the “OASIS Standard Specification” was developed (and published in 2006) for “SOAP messaging” to “provide[] a means to protect a [SOAP] message by encrypting and/or digitally signing a body, a header, or any combination of them (or parts of them)” where “[t]he specified mechanisms can be used to accommodate a wide variety of security models and encryption technologies.”).

209. SOAP's web-based use was well-known to be secure. EX1047 (OWASP - WSS Security Specification), 1-2, 4 ("Transport Confidentiality... Rule - All communication with and between web services", including "SOAP-based Web Services", "containing sensitive features, an authenticated session, or transfer of sensitive data **must** be encrypted using well configured TLS.").

210. REST protocols were also well-known to be secure. EX1048 (OWASP - REST Security Specification), 1-2 ("RESTful web services should use session based authentication" and, "the use of TLS **should be mandated**, particularly where credentials, updates, deletions and any value transactions are performed" using RESTful APIs including because "[t]he **overhead of TLS is negligible** on modern hardware, **with a minor latency increase that is more than compensated by safety for the end user...**").

211. Java's use of security was also well-known. EX1053 (JSSE Guide), 3-5, 68 (Java Secure Socket Extension (JSSE) standard specification providing implementation details for the JSSE API (*i.e.*, secure Java-application-to-Java-application communications (over SSL/TLS sockets), including using Java RMI, between local and remote JVMs over the Internet).

212. Using Java remotely was also well-known to support secure mechanisms. EX1054 (Java RMI Overview), 1 (Java RMI Overview (referenced by EX1053 (Java Secure Socket Extension (JSSE) Reference Guide), 68) identifying

EX1055 (Java RMI Socket Overview) as a “tutorial... discussion of how Java RMI can be used over SSL sockets”); EX1055 (Java RMI Socket Overview), 1 (“Many users are interested in secure communication between Java RMI clients and servers, such as with mutual authentication and encryption”, describes how “[c]ustom socket factories provide a hook for doing this” and references EX1056); EX1056 (Java RMI SSL Overview), 1 (describing how “using Java RMI with SSL, an application can use SSL socket communication... for calls to a remote object” and identifies that “the JDK includes the... JSEE[] API which provides an implementation of SSL sockets.”).

213. It was also well-known to POSITAs that “the OSGi platform is an ideal platform for cloud computing.” EX1008 (Alves), 250, 269 (“The OSGi Platform is an ideal Platform as a Service”), xiii-xiv (“In this age of cloud computing, a system needs to be dynamically adaptable, highly manageable, and easily maintainable. OSGi technology facilitates all this alongside what is generally an extremely light infrastructure footprint.”); EX1011 (Rellermeyer), 3 (“The modularity in OSGi solves a major issue with creating elastic applications for the cloud.”).

## **2. Clouds**

214. As I described in §VI.F above (Technology Overview – Distributed Systems), “clouds” was the name adopted in the mid-2000s for managed private/public “clusters” of computers, and were simply the latest, well-known

evolution of distributed systems. §VI.F above (*citing* EX1063 (Stallings), 699-708; EX1052 (Sosinsky), xxv-xxvi, 19-20, 45, 271-272).

215. POSITAs understood a “cloud” to simply mean a “set of computers accessed remotely”, or “an infrastructure comprising a network of interconnected nodes”, with the “key concept” being that “the user does not know where the server is located, or where there is one server or many; everything is ‘off in the clouds’.” EX1033 (Downing), 94; EX1052 (Sosinsky), xxv, 3, 45; EX1030 (Kashyap), ¶¶0038]; EX1057 (Microsoft Computer Dictionary), 153.

216. Relatedly, POSITAs understood “cloud computing” to simply mean “computing operations carried out on servers that are accessed through the Internet” or “applications and services that run on a distributed network using virtualized resources and accessed by common Internet protocols and networking standards.” EX1033 (Downing), 94; EX1052 (Sosinsky), 3.

217. Cloud deployment models generally included “private”, “community” (shared by several organizations), “public” and “hybrid” (two or more different types of clouds):

**Private cloud:** the cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on-premises or off-premises.

**Community cloud:** the cloud infrastructure is shared by several organizations and supports a specific community that has shared

concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on-premises or off-premises.

**Public cloud:** the cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

**Hybrid cloud:** the cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

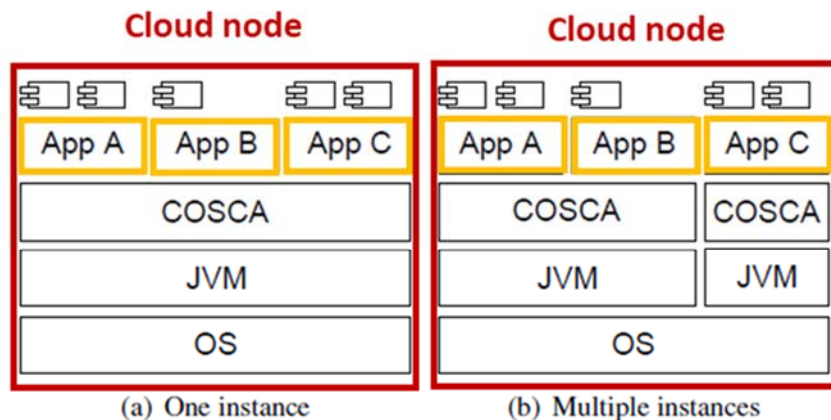
EX1052 (Sosinsky), 3, 7-8; EX1030 (Kashyap), ¶¶[0033]-[0037].

218. The 40+ year-old “virtualization” technology (including as implemented through virtual machines (e.g., JVMs)) was ubiquitous in **cloud servers** by June 2013. *See* §VI.C above (Virtualization) (*citing* EX1031 (Tanenbaum 2008), 569-570; EX1036 (Soltesz), 278-279, Figure 3); EX1052 (Sosinsky), xxvi, 10-12, 19-20, 51-52, 94-95, 100-103.

219. Having a plurality of virtual machines running on a single **cloud server**—*i.e.*, on a single server of a set of servers accessed remotely and/or within a network of interconnected nodes—was well-known to POSITAs long before June 2013. *See, e.g.*, EX1030 (Kashyap), ¶¶[0033]-[0037]; EX1052 (Sosinsky), 69-70, 82; *see also* §VI.C above (Virtualization) (*citing* EX1031 (Tanenbaum 2008), 569-

570; EX1036 (Soltesz), 278-279, Figure 3).

220. For example, Kachele (EX1012) discloses a Java-based, distributed Platform-as-a-Service (PaaS) system “designed to bring OSGi to large scale systems”, in which “[a]pplications... have bundles that are deployed on multiple nodes”, which “provide[s] platform support for elasticity” by “enabl[ing] automatic scale-out and scale-in” of **cloud nodes** based on “monitoring... CPU usage or... memory consumption” of each **node**, and where (as shown in “(b) multiple instances” below right), on any given **cloud node**, “[w]hen an **application** requires higher isolation (e.g. to prioritize **application** execution on OS level), it is... possible to separate an **application** by spawning another dedicated JVM on this **node**” so that there are “multiple instances of the platform running on one **node**”:



**Figure 1.** Deployment and Architecture

EX1012 (Kachele), 19-22, FIG. 1.

221. By June 2013, many other examples of implementations of a single

cloud server running a plurality of virtual machines were well-known, including as depicted and described below.

222. For example, Kawashima (EX1043) discloses that “a **cloud datacenter network**[] consist[s] of many switches, computing nodes, and users’ virtual machines... Figure 1 depicts a **typical** architecture of virtual datacenter network with edge-overlay approach. **Three users’ VMs are running on a same physical host**, and each VM is connected to a same OpenFlow-enabled virtual switch like Open vSwitch...”. EX1043 (Kawashima), 124-125, Figure 1 (reproduced below).

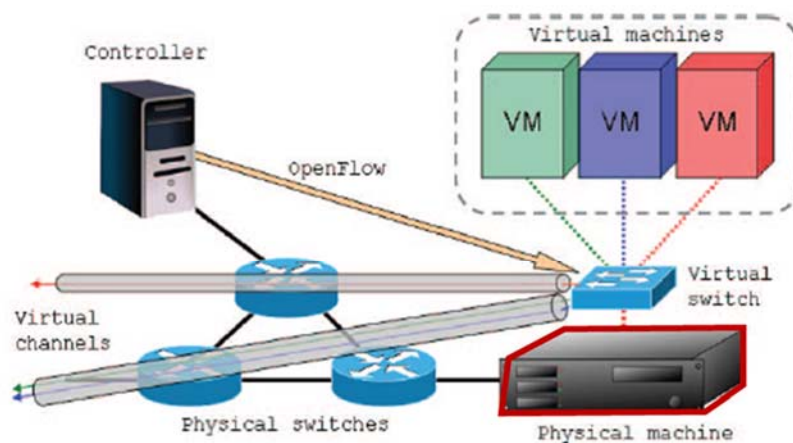


Fig. 1. A typical SDN/OpenFlow-enabled virtual network (edge-overlay)

223. Another example is ExpoNet, as disclosed in EX1039 (Li), which identifies that “the design and implementation of ExpoNet... [is] based on a **cluster of high-performance servers**... As shown in [FIG. 1],... [t]he control plane is software-based solution that provides virtual environment... to serve the data plane. It contains... one physical host machine and **multiple concurrent virtual machines**

that are co-exist [sic] on a single **physical machine**. In our system, we use both OpenVZ and XEN virtualization technologies... according to the choice of the data plane. In the virtual machine, three open source router software components, XORP, Quagga, and Zebra..., provide the routing functionality to generate the forwarding table.” EX1039 (Li), 1-3; *see also* Fig. 1 (which I have annotated below):

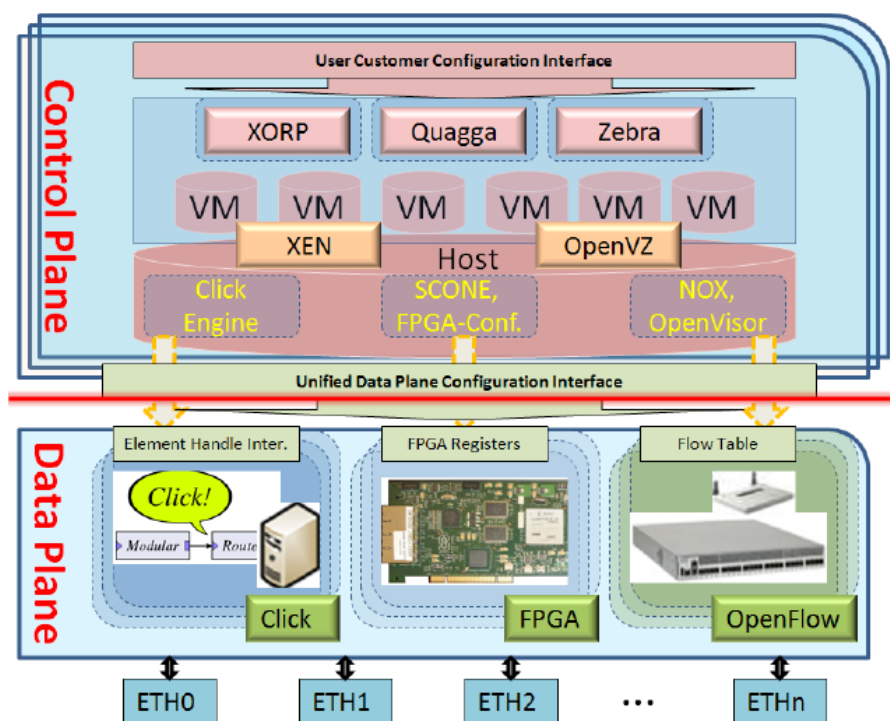


Fig. 1. Overview of ExpoNet Design.

224. Related to the “**infrastructure application marketplace**” recited in the Challenged Claims, it was also well-known to utilize “**application repositories**” (also called “**application stores**”) in cloud-based **distributed systems**, including those implementing OSGi technology (such as the examples that I reproduce and discuss below).

225. For example, Kashyap (EX1030) discloses a **distributed “peak load**

processing environment” including a “local computer system” (e.g., “customer server”) “coupled to [a] remote system” (“the cloud”) via the Internet, where “Linux containers may be used as the virtualization platform for the applications” respectively executing on the local and remote devices, and where the environment also includes “an application store”, into which applications “develop[ed] and test[ed]” by “independent software vendors (ISVs)” are moved “[o]nce tested”, and from which the “local system” and the “remote system” may download respective “instance[s] of the application”, with “the remote instance of the application... be[ing] provisioned” when a “system and/or application utilization” threshold “is reached.” EX1030, ¶[0057], FIG. 4, ¶¶[0070]-[0074], ¶¶[0082]-[0084].

226. As another example, Kim (EX1010) discloses a distributed “smart home software architecture” (depicted in annotated Figure 1 below) built “on top of the Java-based OSGi framework”, in which a “home gateway”, executing OSGi bundles in a JVM running on top of a Linux OS, “connects to a cloud solution center that [i] holds an application store”, which “manages the dependencies of software and home devices, and downloads software drivers and applications to the... home gateway”, and “[ii] provides cloud services... for example, a video surveillance service requiring large amounts of storage” for which software component communications use “Restful web services”:

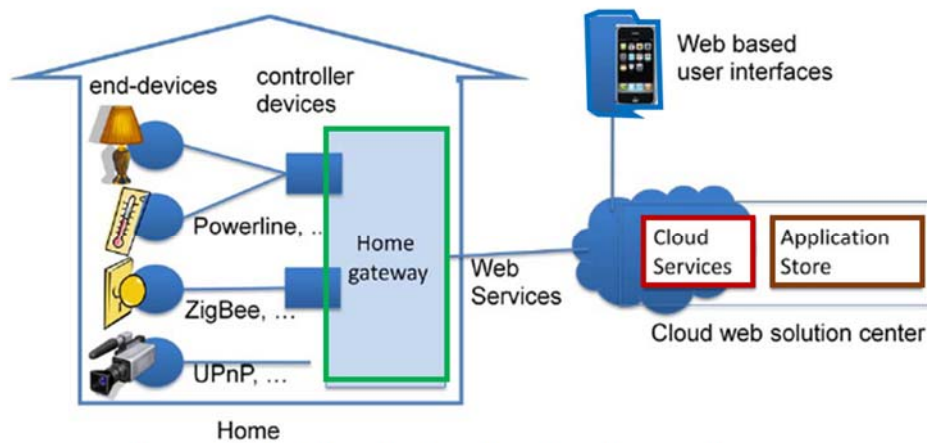


Figure 1: High-level architecture for the smart home

EX1010, 206, 208-209, Figures 1-2; *id.*, 212 (“The **cloud solution center** currently supports the **application store** component, which is deployed in our private cloud.”); *see also* §VI.F.1 above (*citing* EX1042 (Knoplerfish Data Sheets), 1-2).

227. In §VIII.A.2 below and §VIII.A.4 below, I discuss some technical implementation details that the Alves textbook (EX1008) and the Rellermeyer article (EX1011) disclosed regarding OSGi’s Enterprise specification (EX1028).

228. Specifically, Alves (EX1008) is an example of a publication that disclosed technical details for implementing OSGi-compatible distributed systems in various well-known **cloud** environments (including for utilizing a cloud provider’s **OSGi Bundle Repository (OBR)**), and technical implementation details for using well-known protocols (well-known to provide secure communication capability) between **locally** and **remotely** installed software components of distributed **applications** implementing services. *See* §VIII.A.2 below.

229. Moreover, Rellermeyer (EX1011) disclosed a service that, “using

features already provided by OSGi” specifications, has the OSGi management software monitoring and controlling automatic load (resource usage) expansion/contraction across multiple **cloud servers**. See §VIII.A.4 below.

## VII. THE '823 PATENT

230. As I identify in ¶61 above, throughout my Declaration, I use the color scheme that I identify in the table included in ¶61 above, and reproduce again below for ease of reference, to annotate the same elements in the '823 Patent (EX1001) and the prior art references that I discuss herein:

Concept	Color
distributed system for processing data packets comprising a network device and a cloud device	alternating <b>green</b> and <b>maroon</b>
network device	<b>green</b>
cloud device	<b>maroon</b>
distributed application with a component on a network device and a component on a cloud device	alternating <b>pink</b> and <b>orange</b>
network device application that is a component of the distributed application hosted on a network device	<b>pink</b>
cloud application that is a component of the distributed application hosted on a cloud device	<b>orange</b>
application management portal (and associated software)	<b>blue</b>
application repository/store	<b>brown</b>

### A. Summary of the '823 Patent

231. The '823 Patent relates “to a network architecture that facilitates secure and flexible programmability between a user device and across a network with full lifecycle management of services and infrastructure applications.” EX1001 ('823 Patent), 1:14-17.

232. The '823 Patent explains a problem with “existing network solutions” was that network infrastructure equipment— “existing network elements (routers, switches, gateways, cellular base-stations, WiFi access points, etc.)”— used “proprietary hardware and rigid software architecture” respectively provided by vendors. EX1001 ('823 Patent), 1:22-23, 7:45-49.

233. As a result, according to the patent, “Network Operators & Information Technology (NOIT)” could “only configure” such network equipment as allowed by the particular vendor, and could not “add new desired features” to such equipment except “by making such requests” to the particular vendor “or pursuing standardization processes”, which were “time-consuming and resource-intensive.” EX1001 ('823 Patent), 1:22-30; *see id.*, 7:45-49, 8:27-64.

234. The '823 Patent states that the “Network Operators & Information Technology (NOIT)” term “broadly includes carriers, service providers, enterprise networks, and designated/contacted 3rd party administrators.” EX1001 ('823 Patent), 3:55-58.

235. The '823 Patent asserts that the solution to this problem was to “virtualize” various “network features” such that they are programmable as software by NOIT administrators (rather than by the equipment vendors) and “distribute” such virtualized features “across separate network elements that cooperate together to create an advanced, programmable, and scalable network.” EX1001 ('823 Patent), 10:3-6, 7:49-56.

236. The '823 Patent recognizes that this solution required “add[ing] computing power to... networking infrastructure”, and acknowledges that “recent advancements in semiconductor technology” contributed by others, and **not** the '823 Patent’s inventors, is what removed the “cost barrier[s]” associated with this technological requirement and enabled the “programmable platforms in the network and... more flexible network architecture” features of the solution. EX1001 ('823 Patent), 7:49-56.

237. The '823 Patent refers to this solution as a “**distributed software-defined network (dSDN)**” and provides a high-level description of functional requirements and intended results, and a high-level identification of exemplary use cases, while leaving the design and implementation details of the solution software development to the knowledge, skill and creativity of POSITAs. EX1001 ('823 Patent), Abstract; 2:31-40, 8:62-67, 10:61-11:49, 12:27-63, 13:64-16:34, 30:14-38:27 (Table 8).

238. For example, the '823 Patent describes the dSDN as:

[A]n end-to-end architecture that enables secure and flexible programmability across a network with full lifecycle management of services and applications (fxApp). The dSDN also harmonizes fxApp deployment across the network independent of the hardware vendor. As a result, the dSDN simplifies the network deployment lifecycle from concept to design to implementation to decommissioning.

EX1001 ('823 Patent), 7:57-64, Abstract (identical text except it states “(fxDeviceApp)” instead of “(fxApp)” and “application” instead of “fxApp”), 2:31-39 (same as Abstract).

239. FIG. 3 (below, annotated) illustrates “a high-level overview of a dSDN system 300”:

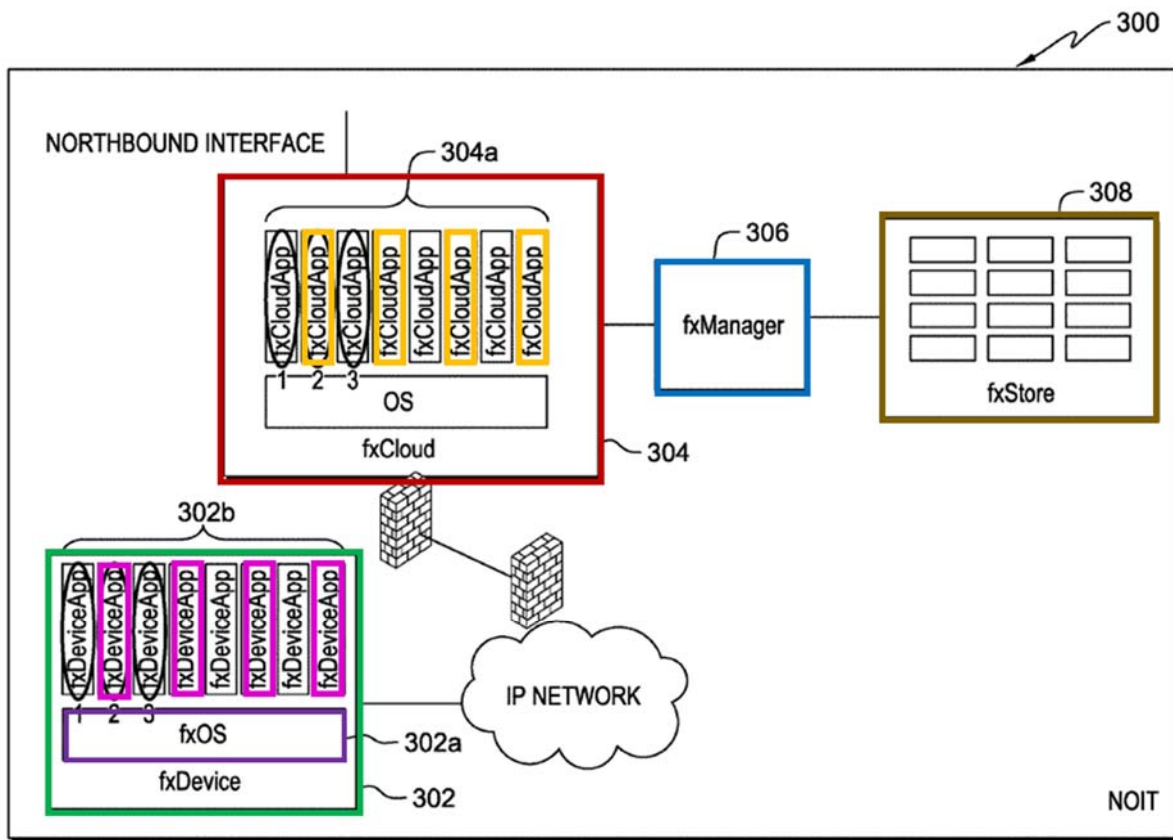


FIG. 3

EX1001 ('823 Patent), 10:8-13, FIG. 3.

240. The exemplary dSDN system 300 includes two hardware devices: a “flexible network device (fxDevice) 302”, and “flexible cloud platform (fxCloud) 304.”

241. The '823 Patent describes the fxDevice as “any networking equipment” that is “powered by a sandboxing operating system named flexible operating system (fxOS).” EX1001 ('823 Patent), 10:13-15.

242. The exemplary dSDN system 300 also includes a “flexible cloud platform (fxCloud) 304”, which the '823 Patent describes as “backend cloud

infrastructure” (e.g., one or more “network servers”) that “interacts **with fxDevice** and **on** other **network elements** on the northbound interface”, and which could be a private, or “shared”/public, cloud platform. EX1001 (’823 Patent), 3:17-19 (added emphasis in bold), 11:32-33 (“network servers”), 24:5 (“fxCloud servers”), 25:36-37 (“cloud servers”), Table 8 (col. 37, #69) (“cloud server”), 17:62-66, 25:3-7.

243. Beyond the hardware components of the **system**, the Patent also described the distributed **applications** (e.g., 1, 2, 3 in FIG. 3 (above)) that run across these **devices**. The Patent explains that each distributed **application** can be separated into a **component** installed on an **fxDevice** (called “**fxDeviceApp**”) and a component installed on an **fxCloud** device (called “**fxCloudApp**”). For example, Figure 12 describes “an example procedure for Application (dApp) Provisioning.” EX1001 (’823 Patent), 18:47-48. The **administrator** identifies a distributed **application** at the start. The system then identifies components of that **application** that should be installed on the **fxDevice** and **fxCloud** device, and installs a respective component on each device. EX1001 (’823 Patent), 19:8-28.

244. The component of the distributed **application** installed on the **fxDevice** is called “**fxDeviceApp**”, and each **fxDevice** “may host several independent and securely isolated applications (on top of fxOS 302a) named **fxDeviceApp 302b**”. EX1001 (’823 Patent), 10:59-61.

245. The component of the distributed **application** installed on the **fxCloud**

device is called an “fxCloudApp.” EX1001 (’823 Patent), 10:25-31.

246. In other words, a distributed application includes an fxCloudApp component “paired with” an fxDeviceApp component. *Id.*; EX1001 (’823 Patent), 3:18-20.

247. As I identify above, the ’823 Patent describes a fxDevice (e.g., router, gateway, etc.) as being “powered by a sandboxing operating system” (e.g., fxOS 302a) that “ensures each [fxDeviceApp] runs as a dedicated process in sure isolation from the other [fxDeviceApps]” running thereon, and that “may be built based on an existing sandboxed operating system (OS) (e.g., Android).” EX1001 (’823 Patent), 10:13-24 (added emphasis in bold), 3:23-24.

248. Relatedly, the ’823 Patent states that an exemplary fxOS 302a includes “[r]untime 424... an embedded virtual machine capable of securely isolating and executing the [fxDeviceApps 302b].” EX1001 (’823 Patent), 11:63-64, 12:22-23, 2:3-4, FIG. 4. The “sandboxing” fxOS is also described as “allow[ing] hot upgrade of the software and applications on the platform with little to no interruption to the operational aspect of the platform and its applications.” EX1001 (’823 Patent), 11:29-32, 19:30-62, FIG. 13.

249. The ’823 Patent describes that the fxOS, and “virtual fabric” between fxDevices and fxCloud in the dSDN, enforce “policies”, “permissions”, and “access levels”, defined for each application (or application “group”) by a network

administrator and/or application developer, which permit and restrict, on a controlled basis, communications, and/or sharing of data, information, and/or “resources” (e.g., memory, CPU cycles, application programming interfaces (APIs), etc.), between (i) **fxDeviceApps** on the same **fxDevice**, (ii) the same, or a different, **fxDeviceApp** on different **fxDevices**, and/or (iii) an **fxDeviceApp** on an **fxDevice** and an “associated” **fxCloudApp** on **fxCloud**. EX1001 (’823 Patent), 11:18-24, 11:53-60, 12:48-58, 14:8-11, 14:25-30, 15:26, 16:6-12,16:27-28, 16:52-67, 18:58-64, 20:50-57, 20:67-21:5, 21:9-31, 21:38-47, 21:51-60, 22:64-23:5, 23:23-30, 24:11-47.

250. The ’823 Patent describes the “Virtual Fabric (fxVF)” (which I annotated gray below), at a fairly high level, as a **dSDN** “service.” The ’823 Patent states that the “Virtual Fabric (fxVF)” “provides an abstraction layer for applications to communicate with each other whether they are in the **fxDevice** or **fxCloud**”, and “enables transparent switching of application and system messages of a dApp between the **fxCloud 304** and the **fxDevice 302** and between different dApps”:

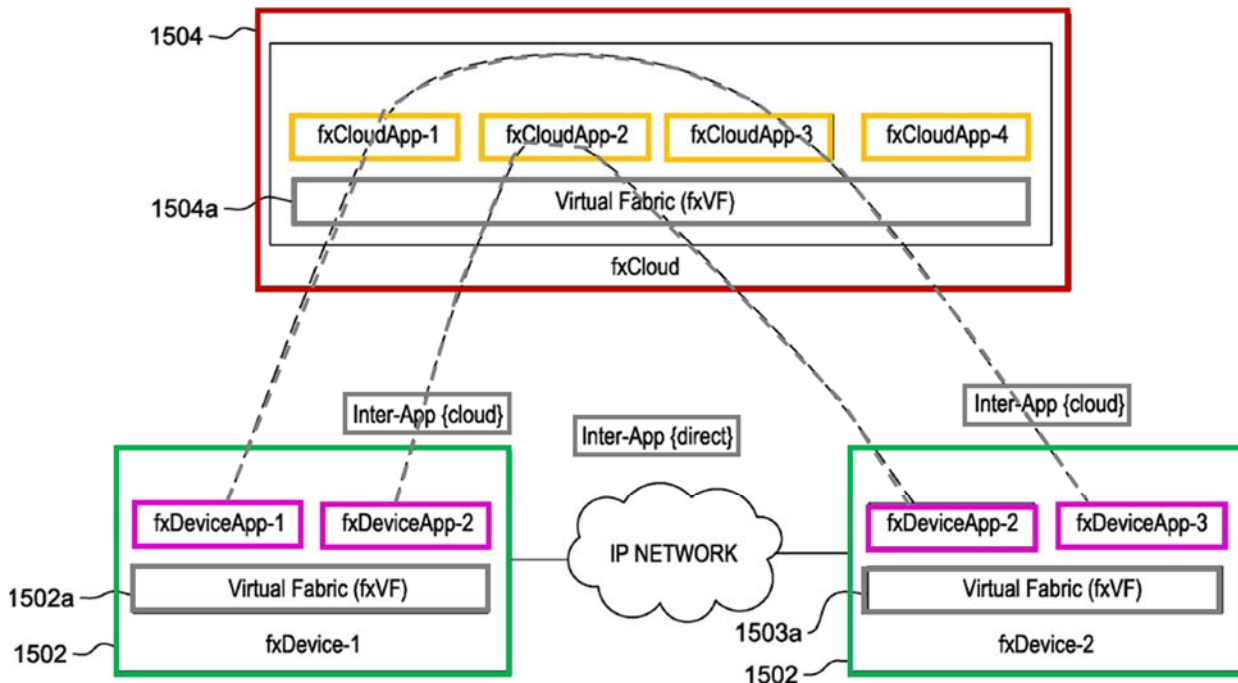


FIG. 15

EX1001 ('823 Patent), 20:35-49, 13:39-42, 2:23, FIG. 15 (annotated above), 11:57-63, 20:50-21:5.

251. The '823 Patent states that fxVF may “provide a secure routing mechanism”, and may use well-known XMPP as “the default protocol”, for such communications between applications (and between application instances). EX1001 ('823 Patent), 20:35-45, 20:50-21:5, 26:19-23.

252. Moreover, the '823 Patent repeatedly emphasizes that “[i]t is important... that the fxDeviceApp 302b and fxCloudApp 304a could use any protocol to communicate with each other” to “allow[] for ultimate freedom for the [application] developers” and “to “allow[] for the system to operate in a loosely

coupled autonomous fashion allowing for asynchronous communication in a distributed fashion.” EX1001 (’823 Patent), 10:40-46, 20:45-48, 24:7-47, 26:65-27:15.

253. In other words, the ’823 Patent uses the term “virtual fabric” broadly. For example, it does not require that the applications communicate over a virtualization overlay network. *Id.*

254. The exemplary **dSDN system** 300 (see FIG. 3, which I reproduced and annotated in ¶239 above) is described as also including an “**application management portal (fxManager)** 306”, and “**an infrastructure application market place (fxStore)** 308”.

255. The ’823 Patent describes that the **fxManager** “presents a user-friendly portal to [a] network administrator” (e.g., “NOIT admin”), through which the administrator may, for example, **manage** the **fxDevice(s)** in the network, **manage** the **fxCloud platform**, perform “[a]pplication lifecycle **management**” tasks (e.g., “provisioning, de-provisioning, upgrade/downgrade applications”, “test [applications] prior to deployment”, etc.), and/or “brows[e] through the certified applications in the **fxStore**.” EX1001 (’823 Patent), 15:10-16:27, 10:47-51, 17:53-19:28.

256. I note that the only other use of the word “certified” in the ’823 Patent specification (other than the claims) is at 10:50-55, where the Patent states: “Using

[fxManager](#) 306, the admin may discover new fxApps from the [fxStore](#) 308, which presents all the tested and certified vApps as well as showing the supported fxOS version, support hardware platforms, and other information such as reviews and number of commercial deployments by all NOIT customers.” EX1002 (’823 Patent), 10:50-55. The ’823 Patent otherwise does not detail what “certified” means, and a POSITA would understand that, given this disclosure in the ’823 Patent, one meaning of “certified”, that is consistent with this word’s plain and ordinary meaning, is certifying that “vApps” were successfully tested.

257. The [fxStore](#) is described as a “fleXible store that presents fxApp to the network”, may be “third party hosted”, contains software applications, and associated updates, developed (and optionally tested) for use in the [dSDN](#) environment by application developers (e.g., NOIT, third party developers/partners) and from which a network administrator may download such applications and updates (e.g., via [fxManager](#)). EX1001 (’823 Patent), 3:31-32, 10:51-55 (“the [fxStore](#) 308 ... presents all the tested and certified vApps as well as showing the supported fxOS version, support hardware platforms, and other information such as reviews and number of commercial deployments by all NOIT customers.”), 13:1-3 (“the [fxDeviceApps 302b](#) ... are written by third party developers or the NOIT”), 15:39-40 (functions of [fxManager](#) include “Uploading applications developed by the NOIT or their partners into the [Application Repository \(AR\)](#)”), 15:48-49 (functions

of **fxManager** include “Browsing through the certified applications in the **fxStore**”), 16:25-35 (“[T]he network administrator could optionally procure the applications a third party hosted application store (**fxStore**). The **fxStore** 308 could potentially categorize the applications several groups ... The **fxStore** 308 could also show the reviews of the advertised applications. In order to maintain quality, the third party **fxStore** 308 may perform rigorous testing of the applications and accompany those results with the advertised application.”); *see also id.* at 16:39-47, 16:59-62, 16:66-17:3, 18:50-51, 19:48-51.

258. The '823 Patent describes that, using well-known digital signature techniques (*e.g.*, X.509 digital certificates, security keys, hashing algorithm, etc.), an application (*e.g.*, binary file in downloadable application package and accompanying manifest file) in the **fxStore** could be “signed” (*e.g.*, by application developer) so that, if a signature match is obtained by the recipient (*e.g.*, **fxManager**) using these same techniques, the application is certified to be authentic and intact (unchanged). EX1001 ('823 Patent), 23:8-11 (“It is important for the target network element to verify the authenticity and integrity of loaded software prior to install. The uniquely defined security keys of the application developers sign the applications.”), 23:21-24:11 (“Each **fxDeviceApp** is accompanied by a manifest that is generated by the Integrated Device Electronics (IDE) at the time of compilation of the **fxDeviceApp** ... In order to ensure that manifest file is actually genuine

manifestation of the `fxDeviceApp` and its integrity remains intact, a hashing algorithm could be used to generate a signature. An example is shown below ... The signature is generated by the IDE/fxSDK at the compile/build phase and is packaged with application binary and the manifest file. Once the distributed `application` package (dAP) is downloaded ... the components of the signature, application binary, and manifest file are unpackaged. The `fxManager` then uses the same hashing algorithm to calculate the signature as above using the pre-shared key (K). If the calculated signature and the unpackaged signature match, it would prove the authenticity of the manifest file and the integrity of both manifest file and the `fxDeviceApp`.”).

259. The patent continues to discuss digital signatures. EX1001 (’823 Patent), 22:55-58 (“Application Developer Identity (ADID): this is a globally unique identity allocated by a globally accessible server and may be included in the developer’s X.509 digital certificate and is used to sign the final developed applications.”), 11:24-29 (“[T]he applications may be signed with unique certificate security keys. The security keys and certificates may be allocated and/or signed by a certificate authority. The `fxDevice` 302 may be protected by validating signed applications to run or to install in `the device`.”), 15:48-49 (the `fxManager` functions may include “[b]rowsing through the certified applications in the `fxStore` and downloading them to the `AR`.”), 16:25-26 (the `fxManager` functions may include

“Certificate and signature verification of the applications”); *see also* §VI.E above (Technology Overview – Security for Software Implemented and Downloaded Over the Internet Including Digital Signature Implementations) and §VI.B.1 above (Technology Overview – Hash Functions and Their Uses).

260. Challenged Claim 18 recites that “the at least one programmable network device and the programmable cloud device are adapted to verify the authenticity and integrity of updates to the plurality of network device applications and the plurality of cloud applications”. *See* §IX below (Claim Element by Claim Element Invalidity Analysis), claim 18 (§IX.C.6 below)).

261. The only disclosure in the '823 Patent related to this function is stated as being performed by a “network device” (e.g., fxDevice) and/or a “cloud device” (e.g., fxCloud server), rather than by the fxManager, is at 11:27-29, 19:14-17 and 19:21-24. EX1001 ('823 Patent), 11:27-29 (“The fxDevice 302 may be protected by validating signed applications to run or to install in the device.”), 19:14-17 (“In step 1207, the fxDevice 1240 verifies the fxDeviceApp package and performs necessary integrity checks of the received software package. Once verified, the fxDeviceApp is installed and re-verified.”), 19:21-24 (“In step 1211, the fxCloud 1250 verifies the fxCloudApp package and performs necessary integrity checks of the received software package. Once verified, the fxCloudApp is installed and re-verified.”).

262. Although the first sentence (EX1001 ('823 Patent), 23:8-9) of the “Software Security Verification” entitled section of the '823 Patent (EX1001 ('823 Patent), 23:6-67) may also be related to Challenged Claim 18, the rest of this section (EX1001 ('823 Patent), 23:10-67) discloses the **fxManager** (and **not** an **fxDevice** or **fxCloud device**) performing signature matching to verify the authenticity and integrity of a downloaded “distributed application package.” EX1001 ('823 Patent), 23:8-9 (“It is important for the target network element to verify the authenticity and integrity loaded software prior to install.”), 23:10-67.

263. The '871 Patent File History explains this further. EX1007 (Parent '871 Patent File History), 135, 138-140 (original '871 Patent claims 1, 15, 17 recited this same “verifies authenticity and integrity” limitation as '823 Patent claim 18). In *id.*, 196, the Examiner rejected these claims as lacking written description support under 35 U.S.C. §112(a) because they “recite[] a desired function of verifying authenticity of applications by the claimed [programmable **network** and programmable **cloud**] devices, but the disclosure fails to sufficiently identify how the function is performed by the devices”, and “[i]nstead the Specification ... discloses that **fxManager** (aka application management portal) ... verifies authenticity of the application.”); *see also, id.*, 252-253, 255-257, 259 (Applicant amended '871 Patent claims 1, 15, 17 to instead recite this same “verifies authenticity and integrity” limitation as being performed by the “**application**

management portal” in response to the Examiner’s written description rejection).

264. The ’823 Patent’s lack of detail in disclosing how, instead of the **fxManager**, an **fxDevice** or an **fxCloud server** would perform signature matching to verify the authenticity and integrity of a respective application received from the **fxManager** is not surprising, given that these digital signature techniques were well-known as were the increased security benefits provided by performing these verification techniques “[b]efore ... install[ing] any [software] patch” and even “before each time it is run.”) *Id.*; Technology Overview §VI.E above (Technology Overview – Security for Software Implemented and Downloaded Over the Internet Including Digital Signature Implementations, ¶¶135-147 above)); Overview of the Hall Textbook §VIII.A.3 below (¶¶349-355 below)).

265. The ’823 Patent describes other exemplary distributed “services” — including the “Distributed Resource Service (dRS)”, the “Distributed Content Service (dCS)”, and the “Distributed Notification Service (dNS)” — that are usable by application developers, and operate based on the “policies”, “permissions” and “access levels” defined for each application (or application “group”) by network administrators and/or application developers and enforced by the **fxVF**, **fxCloud**, and **fxOS** on the **fxDevice(s)** in the **dSDN**. EX1001 (’823 Patent), 17:44-49, 21:38-47, 16:61-67, 21:64-22:6, 21:64-22:7-20, 12:48-55, 7:57-61, 2:31-35.

266. As I identified above, the ’823 Patent describes **dSDN** “resources” as

including, for example, (1) “platform resources” (e.g., “memory size, cache/storage space,... CPU cycles”, etc. of **fxDevice(s)** and **fxCloud server(s)**), and (2) APIs (e.g., intra-app APIs (between **fxDevices**), inter-app APIs (on same **fxDevice**, between **fxDevices**, between **fxDevice** and **fxCloud**), etc.). EX1001 (’823 Patent), 21:9-22, 11:21-24.

267. The ’823 Patent describes an exemplary “Distributed Resource Service (dRS)” as including “software agents residing in the **fxDevice 1604** and **fxCloud 1606**” that, for each running application, “manage [its] access to the platform resources” based on application-specific resource “requirements” and on policies and thresholds defined by a network administrator via **fxManager**. EX1001 (’823 Patent), 18:58-64, 16:61-67, 21:6-47, FIG. 16, 22:11-26 (“Cloud Breathing Procedures”), FIGS. 17A-17B, 21:67-22:5, 14:9-11, 15:64-16:4, Table 8 (#51, #20, #67).

268. The ’823 Patent describes the “Distributed Notification Service (dNS)” as a service “by which the applications could be notified of an event to wakeup and/or respond to the event” such as, for example, notifying an application that a relevant threshold or condition has been reached or exceeded. EX1001 (’823 Patent), 14:15-18, 2:65, 21:62-22:7, Table 8 (#5, #14, #20, #51, #60-#61, #67).

269. The ’823 Patent describes the “Distributed Content Service (dCS)” as a service by which applications (and application instances) share content (e.g.,

application-specific, or system-specific, data or information) with each other based on application-specific API resource requirements and on communication “policy settings and permissions” defined by a network administrator via [fxManager](#). EX1001 (’823 Patent), 21:49-60, 2:64, 12:48-55, 14:19-24, 16:15-16, 28:19-32, Table 8 (#5, #20, #38-#40, #51, #61-#62, #64).

270. Table 8 of the ’823 Patent lists “examples of use cases possible by the... **dSDN**”, including the following examples:

Use	Description
5	Turning on or off APs if nobody is in the office. Sensors in the building send information to <a href="#">fxCloudApp</a> which in turn orders power control of the APs.
9	Caching most visited website or video clip in the enterprise.
10	The most viewed videos and webcasts are cached in the wireless edge
14	Power Calendar: On-demand coverage (triggered by passing traffic thresholds in certain cells or by calendaring): Operators need to design their network for peak usage rates at high costs, leaving their network underutilized most of the time.
20	Application filtering/throttling (secondary conditions: time-based, location-based, subscriber-type, none).
29	IP-PBX implementation in the small-cell or WiFi AP for enterprise use. This would allow a simple out of a box solution. The VoIP phones

Use	Description
	connect to the <b>fxDevice</b> via the LAN connections. The WAN connection could connect to an <b>fxCloudApp</b> for PSTN calls.
38	Integrated smart home with multi-mode femto/WiFi router. For example, the <b>application</b> in the smart home router ( <b>fxDevice</b> ) allows for coordination of home physical security service (such as ADT services). With the <b>fxCloudApp</b> , the user could control the security status of his/her home and remotely control the setting.
39	Home CCTV DVR implementation in an integrated <b>home router</b> . The camera feeds are recorded on the same <b>router</b> caching engine.
40	Enterprise CCTV DVR implementation in an integrated <b>router</b> with <b>cloud</b> backup. The camera feeds are recorded in the <b>cloud</b> .
47	DNS caching at BS to accelerate DNS lookup for subsequent users. This would improve the user experience by reducing response delays.
51	Self adjusting <b>fxCloud</b> compute resources: The <b>fxCloud</b> creates and tears down virtual machines (VMs) based on the traffic load measured by itself or the packet processing and data plane engine. It is also possible that the VM adjustments are made based on the rush hour or pre-configured hours of the day or manually by the administrator.
60	Dynamic advertising in shopping malls. The shopper would use an app from the shopping mall. When the shopper brings up or refreshes the app, the <b>dSDN</b> allows the shopping mall to pinpoint the exact indoor/outdoor location of the user (e.g. the shop the shopper is in) and send special coupon and ads for that store to the shopper.

Use	Description
61	Smart tour guide. The tourist would use an app from the tourism board or museum. When the user brings up or refreshes the app, the <b>dSDN</b> allows to pinpoint the exact location of the tourist (e.g. the painting he/she is looking at) and send tour guide information (e.g. audio, video, hypertext). <b>dSDN</b> allows to localize the positioning and [sic]
62	General hotspot landing page.
64	Power and energy consumption map of the network. A real-time <b>fxCloudApp</b> pulls the power consumption details from each <b>fxDevice</b> periodically or on demand. The pulled data then are combined with the location information and correlated into a map to present a viewable consumption map.
66	High availability. This app allows two or more <b>fxDevice</b> to form a redundancy cluster where they participate in statueful [sic] redundancy and load-balancing. Bandwidth Bursting.
67	Bandwidth Bursting. This feature enables the dynamic adjustment of backhaul or WAN link bandwidth as the demands change in the enterprise or public wireless deployments. In this case, the <b>fxDeviceApp</b> in the <b>fxDevice</b> monitors the WAN link utilization, once a preset threshold percentage has reached, it messages its sister <b>fxCloudApp</b> which in turn would send a request to the <b>fxDeviceApp</b> to increase or decrease the WAN bandwidth. If additional backend reconfiguration is needed, the <b>fxCloudApp</b> performs those additional changes.

EX1001 ('823 Patent), cols. 28-36.

## **B. Prosecution History of the '823 Patent**

271. I have reviewed the file history of the '823 Patent (EX1002) as well as the file history of the earliest parent U.S. Patent No. 9,843,624 (the "Parent '624 Patent"), and the file histories of the two intervening parent U.S. Patent Nos. 10,893,095 and 10,686,871. EX1002 ('823 Patent File History); EX1005 (File History of Earliest Parent '624 Patent); EX1006 (File History of related '095 Patent); EX1007 (File History of related '871 Patent). I summarize certain relevant information that I have reviewed pertaining to these file histories.

272. The '823 Patent was filed on January 6, 2021 as a continuation of, and shares an identical specification with, (1) Application No. 16/900,963 (now U.S. Pat. No. 10,893,095 (the "'095 Patent")) which was filed on June 14, 2020, (2) Application No. 15/836,824 (now U.S. Patent No. 10,686,871 (the "'871 Patent")) which was filed on December 9, 2017, and (3) Application No. 14/295,331 (now the Parent '624 Patent) which was filed on June 4, 2014, which claims priority to U.S. Provisional Application No. 61/834,807 filed on June 13, 2013. EX1001 ('823 Patent), 1-2.

273. As originally filed, the claims of the application that eventually issued as the '823 Patent were identical to the eighteen (18) claims of the '095 Patent that issued on January 12, 2021 (six (6) days after the '823 Patent application was filed). EX1002 ('823 File History), 164-169; EX1040 ('095 Patent), 1; EX1001 ('823

Patent), 1. In response to non-statutory double patenting rejections over the '624 Patent claims and the '871 Patent claims, and a statutory double patenting rejection over the '095 Patent claims, the Applicant filed a terminal disclaimer. EX1002 ('823 File History), 304-311, 332-336.

274. In response to a final rejection indicating that the terminal disclaimer could not overcome the statutory double patenting rejection over the '095 Patent claims, on July 25, 2022, the Applicant canceled the originally filed 18 claims of the '823 Patent application, and added twenty (20) new claims ('823 Patent application claims 19-38) which were virtually identical to twenty (20) claims (amended Parent '624 Patent application claims 1-20) that were, almost six (6) years earlier, presented by the Applicant to a different USPTO examiner during prosecution of the Parent '624 Patent application.

275. Specifically, the July 25, 2022 '823 Patent Application claims 19-38 (EX1002, 374-378) were virtually identical, in the following respective order, to the October 6, 2016 Parent '624 Patent Application claims 1-20 (EX1005, 225-229): '823 Patent Application claims 19-38 were virtually identical to Parent '624 Patent Application claims 1-3, 6, 4-5, and 7-20. EX1002 ('823 File History), 374-378; EX1005 (Parent '624 Patent File History), 225-229.

276. The July 25, 2022 '823 Patent Application claims included independent claim 19 and dependent claim 21 (which I reproduce below):

19. (New) A system comprising a plurality of network elements, said network elements comprising:

a programmable network device adapted to host a plurality of first network applications;

a programmable cloud device adapted to host a plurality of second network applications, wherein the plurality of first network applications in the programmable network device and the plurality of second network applications in the programmable cloud device are in secure communication with each other to form distributed applications; and

wherein the programmable network device and programmable cloud device form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) that allow for simultaneous programming of the programmable network device and programmable cloud device independent of network device hardware and cloud device hardware.

21. (New) The system of claim 19 further comprising:

an application management portal capable of managing life cycles of the distributed applications; and

an infrastructure application market place in communication with the application management portal, said infrastructure market place capable of providing tested and certified distributed applications to the application management portal.

EX1002 ('823 File History), 343-347 (April 8, 2022 Final Office Action), 372-379 (July 25, 2022 Amendment); EX1005 (Parent '624 Patent File History), 222-232

(October 6, 2016 Amendment).

277. During the earlier prosecution of the Parent '624 Patent application, in a December 28, 2016 Office Action, the USPTO examiner rejected the virtually identical claims under 35 U.S.C. 112(b) stating: “It is not clear what ‘simultaneous programming’ of the devices means, because the Specification does not recite ‘simultaneous programming’. Generally, ‘programming’ may refer to software development OR configuration/provisioning of applications ... [I]t is not clear if the claim limitation refers to actual development of software modules or deployment of different software components.” EX1005 (Parent '624 Patent File History), 229-230 (December 28, 2016 Final Office Action).

278. In the same December 28, 2016 Office Action, the USPTO examiner of the Parent '624 Patent application also rejected the virtually identical claims under 35 U.S.C. 112(a), stating that the claimed “(APIs) that allow for simultaneous programming of the programmable network device and programmable cloud device” “fail[ed] to comply with the written description requirement” as this limitation “lacks support in Applicant’s specification”, which the Applicant overcame by deleting the word “simultaneous” from the claim. *Id.*; EX1005 (Parent '624 Patent File History), 242-274 (December 28, 2016 Final Office Action) (original emphasis); *Id.* at 303, 312 (March 28, 2017 Amendment); 324-325 (May 31, 2017 Non-Final Office Action).

279. In the same December 28, 2016 Office Action, the USPTO examiner of the Parent '624 Patent application further rejected these virtually identical claims **based on prior art** and, despite the Applicant making substantial, narrowing claim amendments to attempt to overcome these prior art rejections, twice more rejected the claims **over prior art** until the Applicant agreed to even further narrowing claim amendments in an Examiner's Amendment, with these substantially narrowed claims issuing as the Parent '624 Patent. EX1005 (Parent '624 Patent File History), 250-274 (December 28, 2016 Final Office Action), 302-314 (March 28, 2017 Amendment), 320-347 (May 31, 2017 Non-Final Office Action), 423-438 (August 31, 2017 Amendment), 445-471 (October 4, 2017 Notice of Allowance including Examiner's prior art rejections of Applicant's August 31, 2017 Amended Claims, Interview Summary, and subsequent Examiner's Amendment).

280. In contrast, during the prosecution of the '823 Patent application, the different USPTO examiner only rejected the virtually identical claims (*see* ¶276 above) under 35 U.S.C. 112(a) "as failing to comply with the written description requirement", stating "[p]articularly Applicant's specification is completely silent as to the steps or processes for the *'simultaneous programming of the programmable network device and programmable cloud device independent of network device hardware and cloud device hardware'*", but stating that these claims "**are allowed over the prior art.**" EX1002 ('823 File History), 372-379 (July 25, 2022)

Amendment), 382-390 (September 13, 2022 Non-Final Office Action) (bold emphasis added), 408-416 (September 19, 2022 Amendment), 418-426 (December 16, 2022 Final Office Action), 444-445 (March 20, 2023 Examiner Interview Summary), 455-464 (April 9, 2023 Amendment, deleting the word “simultaneously” from independent claims 19, 39).

281. In its December 16, 2022 rejection, the different USPTO Examiner also stated “Applicant’s specification does not appear to describe specific steps for programming the network device applications and plurality of cloud applications by the plurality of upper layer application programming interfaces (API). Thus, mentioning that steps of the invention can broadly be performed ‘*simultaneously*’, as in ¶0030, is insufficient to support the specific [claimed] limitation” (see ¶276 above). EX1002 (’823 File History), 420-421.

282. I also note that the same USPTO Examiner of the ’823 Patent application (Brendan Higa), upon taking over examination of the related ’871 Patent application from the USPTO Examiner that prosecuted the Parent ’624 Patent application (Raji Krishnan), only issued double patenting rejections of the claims of the related ’871 Patent application and of the claims of the related ’095 Patent application, which the Applicant overcame by simply filing terminal disclaimers. EX1007 (’871 Patent File History), 263-270, 309-310, 325; EX1006 (’095 Patent File History), 115-119, 120-129.

283. Below I show the cumulative amendments that the Applicant made, in response to these written description rejections (and without receiving any prior art rejections), to claims 19 and 21 of the '823 Patent application (which ultimately issued as claims 1 and 3 of the '823 Patent), and in which I have (1) added italicized, bracketed notations corresponding to the issued '823 Patent claims, and (2) emphasized (with italics) the Applicant's deletion of the word "simultaneously" in claim element [1.4], which the Applicant had added in a September 2022 claim amendment, but deleted in an April 2023 amendment to overcome the Examiner's written description final rejection of the claims:

*[1.0]* A system comprising ~~a plurality of network elements, said network elements comprising:~~

*[1.1]* a programmable network device adapted to host a plurality of [first] network device applications;

*[1.2]* a programmable cloud device adapted to host a plurality of [second network] cloud applications,

*[1.3]* wherein the plurality of [first] network device applications ~~in the programmable network device~~ and the plurality of [second network] cloud applications ~~in the programmable cloud device~~ are in secure communication with each other to form distributed applications; and

*[1.4]* wherein the ~~programmable~~ plurality of network device applications and ~~programmable~~ plurality of cloud applications device form unified capabilities enabling a plurality of upper layer application

programming interfaces (APIs) [that] ~~to allow for simultaneous programming of *simultaneously* program the programmable plurality~~ of network device applications and ~~programmable plurality of~~ cloud [device] applications independent of network device hardware and cloud device hardware.

[3.0] The system of claim 1 further comprising:

[3.1] an application management portal capable of managing life cycles of the distributed applications; and

[3.2] an infrastructure application ~~market place~~ marketplace in communication with the application management portal, said infrastructure ~~market place~~ marketplace capable of providing tested and certified distributed applications to the application management portal.

EX1002 ('823 File History), 410-414 (September 19, 2022 Amendment), 418-426 (December 16, 2022 Final Office Action), 444-445 (March 20, 2023 Interview Summary), 457-461 (April 17, 2023 Amendment).

284. I have been instructed by counsel to assume that the “wherein the plurality of network device applications and plurality of cloud applications **device** form ...” (in claim element [1.4]) is a typographical error and to also interpret this claim limitation as reciting “wherein the plurality of network device applications and plurality of cloud applications form...”. I note that, in a September 19, 2022 Amendment (EX1002, 410), the Applicant used strikethrough to indicate its deletion of the word “device” in this claim element (“... programmable plurality of cloud

applications device ...”) but, in an April 17, 2023 Amendment (EX1002, 457), the word “device” [re]appeared in this claim element (“... plurality of cloud applications device ....”).

285. I also note that claim element [19.4] recites “wherein the plurality of network device applications and plurality of cloud applications form...”, *i.e.*, without the word “device” between “cloud applications” and “form”. *See* EX1002 (’823 File History), 410, 457; EX1001 (’823 Patent), claims 1, 19.

286. I also note that, in a September 19, 2022 Amendment (EX1002, ’823 File History, 410) during prosecution of the ’823 Patent application, the Applicant used strikethrough to indicate its deletion of the words “and certified” in claim element [3.2] (“... capable of providing tested and certified distributed applications ...”) but, in an April 17, 2023 Amendment (EX1002, ’823 File History, 457), the words “and certified” [re]appeared in this claim element (“... capable of providing tested and certified distributed applications ...”). I also note that claim element [19.4] does not include the words “and certified” and instead recites “... capable of providing tested distributed applications ...”. *See* EX1002 (’823 File History), 410, 457; EX1001 (’823 Patent), claims 3, 19.

287. In response to the Applicant’s April 17, 2023 Amendment, in which it deleted the word “simultaneously” from independent claims 19 and 39 (which issued as independent claims 1 and 19, respectively), the Patent Office issued a Notice of

Allowance on May 9, 2023, and issued the '823 Patent on July 4, 2023. EX1002 ('823 File History), 457, 461, 466-471, 495-498; EX1001 ('823 Patent), 1.

### **C. Priority Date of the Challenged Claims**

288. The application that issued as the '823 Patent was filed on January 6, 2021, and claims priority, as a continuation, to U.S. Patent No. 10,893,095 (the '095 Patent), to U.S. Patent No. 10,686,871 (the '871 Patent), to U.S. Patent No. 9,843,624 (the Parent '624 Patent), and to U.S. Provisional Application No. 61/834,807 filed June 13, 2013. EX1001 ('823 Patent), 1-2.

289. I have been instructed by counsel to assume, for the purpose of this *inter partes* review proceeding, that the priority date of claims 1-5, 7-8, 12-13 and 15-19 of the '823 Patent is the June 13, 2013 filing date of U.S. Provisional Application No. 61/834,807 ("Priority Date").

## **VIII. OVERVIEW OF THE GROUNDS FOR CHALLENGE**

290. As I describe in detail below, in my opinion, a POSITA would have found the invention of Claims 1-2, 12-15 and 19 of the '823 Patent obvious over the teachings of Vasell (EX1004/EX1015/EX1016, *see* §VIII.A.1 below), in view of the teachings of Alves (EX1008, *see* §VIII.A.2 below) and Rellermeyer (EX1011, *see* §VIII.A.4 below).

291. As I also describe in detail below, in my opinion, a POSITA would have found the invention of Claims 3-5, 7-8 and 18 of the '823 Patent obvious over the

teachings of Vasell, in view of the teachings of Alves (EX1008, *see* §VIII.A.2 below), Rellermeier (EX1011, *see* §VIII.A.4 below), and Hall (EX1009, *see* §VIII.A.3 below).

292. Throughout my Declaration, I may collectively refer to Claims 1-5, 7-8, 12-13, 15 and 17-19 of the '823 Patent as the “Challenged Claims”.

**A. Overview of the Prior Art in the Grounds for Challenge**

293. As I identify in ¶61 above, throughout my Declaration, I use the color scheme that I identify in the table included in ¶61 above, and reproduce again below for ease of reference, to annotate the same concepts in the '823 Patent (EX1001) and the prior art references that I discuss herein:

Concept	Color
distributed system for processing data packets comprising a network device and a cloud device	alternating <b>green</b> and <b>maroon</b>
network device	<b>green</b>
cloud device	<b>maroon</b>
distributed application with a component on a network device and a component on a cloud device	alternating <b>pink</b> and <b>orange</b>
network device application that is a component of the distributed application hosted on a network device	<b>pink</b>
cloud application that is a component of the distributed application hosted on a cloud device	<b>orange</b>
application management portal (and associated software)	<b>blue</b>
application repository/store	<b>brown</b>

**1. Vasell (EX1004/EX1015/EX1016)**

294. In my Declaration, I refer to the combined disclosure of EX1004 (Vasell Patent), with two that it incorporates by reference: EX1015 (Vasell 1<sup>st</sup> Provisional) and EX1016 (Vasell 2<sup>nd</sup> Provisional) as “Vasell”. I make each citation in my Declaration to the corresponding Vasell document.

295. The Vasell Patent (EX1004) is U.S. Patent No. 6,496,575, titled “Application and Communication Platform for Connectivity Based Services”, issued December 17, 2002 from Application No. 09/326,550 filed June 7, 1999, named “Gatespace AB” as its Assignee, named Jesper Vasell, Tom Idermark, Malte Lilliestråle, Hans Thorsen, Staffan Truvé, Carlo Pompilii, Johan Ljungberg, and Jorgen Andersson as inventors, and claimed priority to, and incorporated by reference in their entirety, EX1015 (Vasell 1<sup>st</sup> Provisional) and EX1016 (Vasell 2<sup>nd</sup> Provisional). EX1004 (Vasell Patent), 1, 1:6-10; *see also* ¶177 above.

296. The Vasell 1<sup>st</sup> Provisional (EX1015) is U.S. Provisional Patent Application No. 60/088,437, is titled “Application and Communication Platform for Connectivity Based Services”, was filed June 8, 1998, and named Malte Lilliestråle, Hans Thorsen, Jesper Vasell, Staffan Truvé, Carlo Pompilii, Johan Ljungberg, Jorgen Andersson as inventors. EX1015 (Vasell 1<sup>st</sup> Provisional), 1-2; *see also* ¶179 above.

297. The Vasell 2<sup>nd</sup> Provisional (EX1016) is U.S. Provisional Application

No. 60/123,971, is titled “E-Box System: An Electronics Services Enabler”, was filed March 12, 1999, and named Malte Lilliestråle, Tom Idermark, and Jesper Vasell as inventors. EX1016 (“Vasell 2<sup>nd</sup> Provisional”), 1-3; *see also* ¶177 above.

298. As I identify in §III above (Materials Considered), EX1016 (“Vasell 2<sup>nd</sup> Provisional”) includes a copy of the article published as Idermark, T., Lilliestråle, M., & Vasell, J., “Ericsson’s e-box system—An electronic services enabler”, *Ericsson Review*, (1), 1999, 38-44, as confirmed by Ericsson’s website archived, for example, on March 3, 2000 at Internet Archive’s Wayback Machine at [https://web.archive.org/web/20000303195310/http://www.ericsson.se/review/issue\\_s.taf](https://web.archive.org/web/20000303195310/http://www.ericsson.se/review/issue_s.taf) and, for example, on February 16, 2003 and November 27, 2004 at Internet Archive’s Wayback Machine at [https://web.archive.org/web/20030216191220/http://www.ericsson.com/about/publications/review/1999\\_01/23.shtml](https://web.archive.org/web/20030216191220/http://www.ericsson.com/about/publications/review/1999_01/23.shtml) and [https://web.archive.org/web/20041127035720/http://www.ericsson.com/about/publications/review/1999\\_01/files/1999015.pdf](https://web.archive.org/web/20041127035720/http://www.ericsson.com/about/publications/review/1999_01/files/1999015.pdf), respectively.

299. As I further discuss in §VI.F.1 above, Vasell discloses foundational concepts of a **distributed system** architecture, including implementation in a Java run-time environment, that were developed in a three-year joint research project in the late 1990s between Swedish companies Ericsson and CR&T, where the named inventors worked at that time, and which were eventually standardized under the

Open Standards Gateway Initiative (OSGi), including its exemplary implementation in a Java run-time environment.

300. As I discuss in detail in ¶¶301-334 below of this §VIII.A.1 below, Vasell, like the '823 Patent (*see* §VII.A above), discloses a system for processing data packets having **devices** spread across the Internet, called a “**service gateway system**.” Vasell discloses a “**network device**” called a “**service platform server**,” and a “**cloud device**,” called “**service provider equipment**” or a “**network operator server**.” Vasell also discloses a plurality of distributed **software applications**, each having a **component** installed on a **network device** and a component installed on a **cloud device** (and each implementing a service for end users). Moreover, Vasell teaches a **management service** having an **external interface** to manage the lifecycle of the distributed **applications** and manage upgrades of the components of the distributed **applications**. As I also discuss in more detail in ¶¶301-334 below of this §VIII.A.1, although Vasell uses different words (and, in many cases, more detail) than the '871 Patent, Vasell otherwise describes the identical concepts.

301. Specifically, Vasell discloses a “**service gateway system**” that “provides a platform for connectivity-based services”, where “[e]ach service may comprise a set of functionalities and logic which are implemented as **software service applications**”, and where each “**service application** may be **distributed** among various pieces of equipment which are geographically separated.” EX1004 (Vasell

Patent), 4:31-50, 2:64-3:6, 1:41-45, 8:53-58; EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Services should be implemented as **distributed applications**. By executing over several infrastructure nodes, more complex services can be developed without the need for extremely powerful and complex **edge servers**. **Distributed applications** will considerably increase the technical life and reduce the cost of the **edge servers**.”), 5 (Figure 2), 9 (Figure 5, “Box D, Distributed Services System Architecture” including, *e.g.*, “The local component of the service is implemented using Java **applications** that run on the **edge server** (and optionally, in the management system [“a key tool for the **NO** [**network operator**]”] and at the site of the **SP** [**service provider**].”).

302. Vasell identifies an “illustrative” example of “the **service gateway system**” as “the entire system shown in FIG. 2”, which shows, as I have annotated below (*see* my explanation of my annotations in ¶293 above and ¶304 below) and refer to throughout my Declaration as “Annotated FIG. 2”, a “**service platform server 22**” “connected to the local area network 10” and “connected to [a] **network operator server 24** via the Internet”, which is connected to a plurality of “**service provider equipment 34** via... Internet 26”:

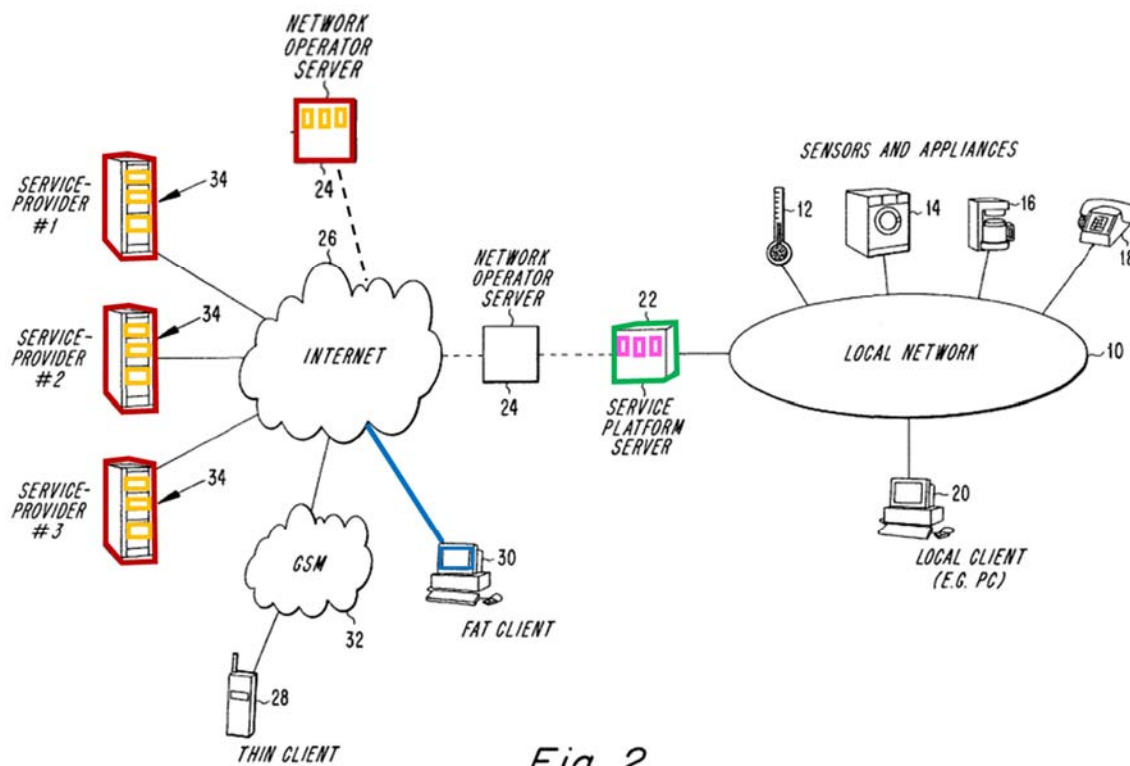


Fig. 2

EX1004 (Vasell Patent), FIG. 2, 4:31-47, 5:61-6:20, 6:62-7:14.

303. Vasell further explains these services. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Services should be implemented as **distributed applications** ...”), 5 (Figure 2 (which I have annotated (similar to Annotated FIG. 2 above) and reproduced below)), 6 (“the *e-box* operator is the owner of the *e-box* network and, as such, is responsible for managing the individual units. From a technical viewpoint, this network is an operation and maintenance system.”), 7 (description of “system software”), 8 (description of “**service applications**” and “system services”), 9 (Figure 5, “*Network operator* (NO): The entity that operates and maintains the system service registry and network of service access points and edge servers. The

NO sells network access to service providers and can have roaming agreements with other NOs to provide a global service network. Examples of potential system service providers include network operators and Internet service providers.”).

304. I have annotated FIG. 2 (above) to show, as disclosed in embodiments in Vasell, (i) a plurality of **service applications** (each comprising one or more software components, each a “boxlet within [a] cell”) hosted on a respective Java Virtual Machine (JVM) running on the respective OSs of the **service platform server(s) 22**, **network operator server(s) 24** (different connection options also illustrated (EX1004, 6:65-7:3)), and the **service provider servers 34**. *Id.*; EX1004 (Vasell Patent), 2:64-3:1, 8:53-9:18 (*see* ¶314 below), 24:49-51 (“wherein said **service application** is a **distributed** software, that is **distributed** across **processors** in addition to said **service gateway unit**”), 12:27-29 (“the **software elements** of the **service gateway system** may be **distributed** among various units of geographically separated equipment, as shown in FIG. 2”), 11:13-58 (*see* ¶¶316-317 below), 18:23-30 (disclosing “**local**” and “**remote**” devices using a “Java Virtual Machine (JVM), or the like” or “multiple JVMs”).

305. Vasell 2<sup>nd</sup> Provisional (EX1016) adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (description of the Java “[d]evelopment environment for **application software**” and that “[s]ervices should be implemented as **distributed applications** ...”), 5 (Figure 2: “The Ericsson **e-service infrastructure**” (which I have annotated

(similar to Annotated FIG. 2 above) and reproduced below), 7 (“The system software consists of: • an operating system; ... • server components—for example, Web servers ... • a Java runtime in the form of a Java virtual machine (JVM). Any UNIX-like standard system, including Open Source, can be chosen as an operating system.”), 8 (description of “service applications” and “system services”), 9 (Figure 5, description of “Edge server (ES)”, “Management system (MS)”, “network operator (NO)”, “Service provider (SP)”); see also §IX.B.1 below (elements [1.1]-[1.4] below).

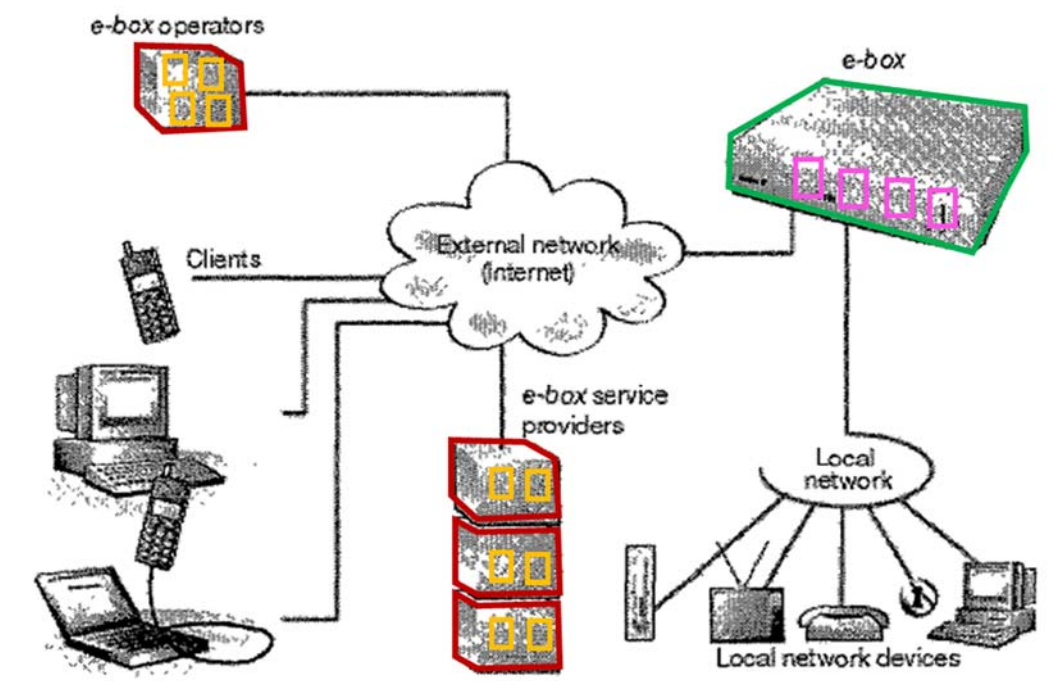


Figure 2  
The Ericsson e-service infrastructure.

306. Vasell describes the “service platform server 22” as “an edge server” and “the hardware portion of a service gateway or services platform”, and states it “may provide service support for” “household users and residences..., an entire

office building, an industrial plant, or even a campus-type organizational facility.” EX1004 (Vasell Patent), 5:61-6:8, 10:32-41. Vasell also states that the “**service platform server 22**” is “logically considered as a node associated with the network operator rather than the local area network [10] since... control for [its] maintenance, operation and support... is preferably performed by the network operator.” EX1004 (Vasell Patent), 6:9-20, 6:45-49.

307. Vasell states that “**network operator server 24 equipment** may include one or more servers, i.e., relatively powerful computers operating in conjunction with one or more secondary storage devices”, and “handle[d] control for the operation, maintenance and support of the service[s] implementation” and “of the **service platform servers 22** of various end users.” EX1004 (Vasell Patent), 6:45-49, 6:58-7:3, 6:16-20, 9:25-40 (describing examples of such maintenance and support software), 11:40-46 (“[a]lthough frequently it will be desirable to interpose an intermediary network operator, there may be certain embodiments of the present invention where the network operator and the associated equipment may be eliminated” and **service provider servers 34** would instead provide such maintenance and support functions).

308. Vasell also states that “network operators using the **network operator servers 24** may act as intermediaries between service providers and the end users of those services.” EX1004 (Vasell Patent), 6:45-7:3, 11:40-46, 6:54-58 (“Various

corporate entities may function as network operators according to the present invention, including, for example, communication utilities such as public telephone companies, communication companies, etc.”); EX1016 (Vasell 2<sup>nd</sup> Provisional), 6 (“the *e-box* operator is the owner of the *e-box* network and, as such, is responsible for **managing** the individual units. From a technical viewpoint, this network is an operation and maintenance system.”), 7 (description of “system software”), 8 (description of “system services” and “**service application management**”), 9 (Figure 5, “*Network operator* (NO): The entity that operates and maintains the system service registry and network of service access points and edge servers. The NO sells network access to service providers and can have roaming agreements with other NOs to provide a global service network. Examples of potential system service providers include network operators and Internet service providers.”; “*Management system* (MS): The management system is a key tool for the NO. It allows the NO to control edge servers and the **service applications** that run on them.”).

309. Vasell states that “**service provider equipment 34**, which may include computer systems or servers, may be the initial repository for **software** used to implement connectivity based services”, and identifies such “services are implemented upon request.” EX1004 (Vasell Patent), 7:4-20, 12:31-33; EX1016 (Vasell 2<sup>nd</sup> Provisional), 9 (description of “*Service Provider* (SP)”).

310. Vasell emphasizes that, “[b]ecause the **service gateway system** is not

limited to a particular technology or protocol, it can evolve freely with respect to access technology.” EX1004 (Vasell Patent), 7:12-14.

311. Based on Vasell’s disclosure that I discuss above, in my opinion, POSITAs would understand Vasell’s disclosed network of **network operator server(s) 24 and service provider servers 34**, accessed “transparent[ly]” through the Internet, each hosting respective **service application** components on a respective JVM running on the respective server OS to implement services through the distributed **service gateway system**, to constitute a **cloud**, and, as I explain in §IX.B.1 below (claim element [1.2] below), each such **server** is a programmable **cloud device**. *See also* EX1004 (Vasell Patent), 2:60-3:1 (“the ... implementation ... of services ... is transparent to both” “different service providers and the end user” and “[e]ach service may comprise a set of functionalities and logic which are implemented as software **service applications**.”); Technology Background §VI.F.2 above (Distributed Systems - Cloud) (*citing* EX1033 (Downing), 94; EX1052 (Cloud Computing Bible), xxv, 3, 45-46, 271-272; EX1030 (Kashyap), ¶[0038]; EX1057 (Microsoft Computer Dictionary), 153).

312. Regarding the implementation of services through the **service gateway system**, Vasell discloses that “implementation of connectivity based services according to the present invention may take advantage of **distributed processing techniques** whereby a service may be implemented using **software** operating on two

or more of the servers or processors associated with the [service gateway system] architecture illustrated in FIG. 2.” EX1004 (Vasell Patent), 8:53-58, 2:60-3:1 (“[e]ach service... comprise[s] a set of functionalities and logic which are implemented as software service applications.”).

313. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Services should be implemented as distributed applications. By executing over several infrastructure nodes, more complex services can be developed without the need for extremely powerful and complex edge servers. Distributed applications will considerably increase the technical life and reduce the cost of the edge servers.”), 5 (Figure 2), 7 (“A service application typically fulfills the role of a gateway between servers in the external network (from which the service is accessed and controlled) and devices in the local network.”), 9 (Figure 5, “Box D, Distributed Services System Architecture” including, *e.g.*, “The local component of the service is implemented using Java applications that run on the edge server (and optionally, in the management system [“a key tool for the NO [network operator]”] and at the site of the SP [service provider].”).

314. As I identified above and confirm with examples below, Vasell discloses a plurality of distributed software applications—called “service applications”, each having a component installed on a service platform server and a component installed on a network operator server and/or service provider

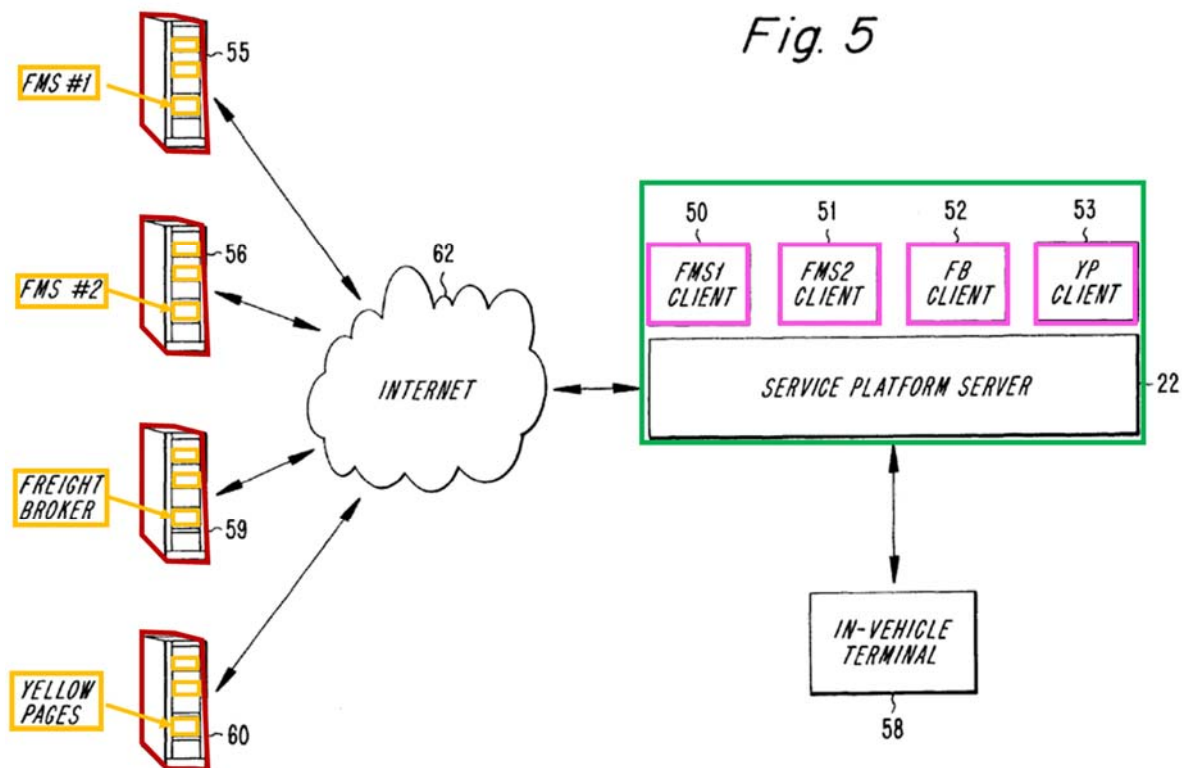
equipment, and each implementing a “service” for end users. End users ranged from “household users and residences..., an entire office building, an industrial plant, or even a campus-type organizational facility.” EX1004 (Vasell Patent), 10:32-41.

315. Referring to Annotated FIG. 2, Vasell discloses an example of “distributed processing” of service applications (each including a plurality of distributed components) in the service gateway system:

For example, a first portion of software associated with the environmental service described above may reside permanently on the #1 service provider equipment 34. A second portion of the software used to implement environment control in the end user’s residence may reside as part of the network operator server 24. Likewise, a third portion of the software used to implement this service may reside on the service platform server 22. The distribution of software used to implement any particular service will, of course, depend upon the nature of the service, the nature of the software, and the bandwidth associated with various information transfer needed to implement the desired service... Rather than being distributed over all three pieces of equipment, those skilled in the art will appreciate that software may be distributed between the network operator server 24 and the service platform server 22, or between the service provider equipment 34 and the service platform server 22, or any other combination of processors deemed desirable.

EX1004 (Vasell Patent), 8:53-9:18, Annotated FIG. 2 (in ¶¶302-304 above), 24:49-51, 12:27-29, 11:42-48; EX1016 (Vasell 2<sup>nd</sup> Provisional), 7-9, Figure 2, Figure 5.

316. Vasell discloses another example of distributing service application software components, with a component at a client location and another component at a service provider location, which is illustrated in Annotated FIG. 5 (below), and in which, using a trucking industry example, mobile service platform server 22 (“outfitted” on a truck) has various “service software client packages” (50-53) loaded thereon (e.g., in a JVM running on the device OS, see ¶304 above) that communicate with associated software components loaded on service provider servers (55, 56, 59, 60) (e.g., in respective JVMs running on each device OS, see ¶304 above) to provide services to trucking industry users:



EX1004 (Vasell Patent), 11:11-20, 11:34-38, Annotated FIG. 5; *see also* ¶304 above (*citing* EX1004 (Vasell Patent), 18:27-30; EX1016 (Vasell 2<sup>nd</sup> Provisional), 4, 7-9).

317. For example, “two different freight companies, FMS #1 and FMS #2,” may each use **mobile service platform server 22** for services by having respective **software** on respective **service provider servers (55, 56, 59, 60)** communicate with **associated software packages (50-53)** running on the **mobile server** to “coordinate operations of the truck” and for inter-application information exchanges (*e.g.*, “**client software package**” may provide **freight company server (55, 56) software** with “information regarding the current location of the truck and its anticipated arrival time of delivery”; “**freight broker**” 59, and/or “**yellow pages**” 60, **server software** may provide information required by “**client software packages**”; etc.). *Id.*; EX1004 (Vasell Patent), 11:17-39; *see also id.*, 11:50-58 (describing an example of **service provider equipment** providing a plurality of services: *e.g.*, a “utility compan[y]” **server** (*see* ¶309 above) “implementing connectivity based services including automated meter reading, real time pricing ..., automated load control ...”), 2:64-66 (“Each service may comprise a set of functionalities and logic which are implemented as **software service applications.**”).

318. Vasell emphasizes the importance that communications between the respective distributed **service application** components (“portions”/“boxlets”), each hosted on respective JVMs running on the respective OSs of the **local** and **remote**

devices of the **service gateway system** (see ¶¶304 above), “**be secure**” and, therefore, expressly “envision[ed] **widespread usage of... security techniques including data encryption**, various authorization and identity verification techniques, etc. to be provided **so that the implementation, monitoring and billing associated with these services is robustly defended against intrusion and manipulation.**” EX1004 (Vasell Patent), 11:64-12:10; see ¶¶304-306 above (citing EX1004 (Vasell Patent), 18:27-30; EX1016 (Vasell 2<sup>nd</sup> Provisional), 4, 7-9); see also EX1016 (Vasell 2<sup>nd</sup> Provisional), 3 (“Consumer Requirements of E-Services” include “*Security*: Consumers **do not tolerate** invasion of privacy or hacking into mission-critical services by strangers or neighbors.”), 6 (“*Security*: The nature of many of the services that can be expected to be based on the *e-box* system, and the fact that the system will be shared by multiple service providers, **make security a major issue**. Mechanisms for authentication and authorization **must** be well integrated into the *e-box* system design.”).

319. Vasell also discloses that, because “the **service gateway system** is accessed by and shared with multiple service providers and service gateway users”, “mechanisms for **authentication, authorization** and access control are integrated within the **service gateway system** design.” EX1004 (Vasell Patent), 4:52-65, 15:46-47; see also EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Service Provider Requirements of an **E-Service Infrastructure**: ... *Integrity*: Different service providers must be

allowed to share the **infrastructure** without adversely affecting each other's services."), 6 ("Robustness ... The *e-box* must also be robust with respect to errors in **software** developed by different service providers; that is, two **applications** must not interfere with one another. This latter property is referred to as *application integrity*."), 7 ("System software ... The system must have fairly strong resource-control and protection mechanisms to ensure **application** integrity."), 8 ("There is a strict requirement for integrity between **applications**, meaning that two **applications** should not be able to inadvertently interact in such a way as to cause one or both of them to fail."), 9 (identifying the disclosed "new **e-service infrastructure**" as "an open platform that can be used by several independent service providers" and, accordingly, "[i]n the design, much effort has gone into **ensuring system security, robustness** and remote-management capability.").

320. Unsurprisingly, Vasell does not disclose the details of implementing these well-known security techniques (*see* ¶¶318-319 above) in the **service gateway system**. For example, Vasell does not explicitly state that these well-known authentication, authorization, and/or identity verification security techniques are used for the downloaded **service applications** (including updates), or involve the use of unique security keys associated with the **service applications**, *e.g.*, using a public key of the application developer to verify the authenticity and integrity of the digitally signed application update.

321. However, as I explained in my Technology Overview sections §VI.B above, §VI.B.1 above, and §VI.E above, in my Overview of the Prior Art in the Grounds for Challenge – The Hall Textbook (EX1009) §VIII.A.3 below, and in my Motivation to Combine the Teachings of the Prior Art in the Grounds for Challenge §IX.A below, POSITAs would recognize that Vasell does not go into such detail because such digital signature matching techniques were well-known to POSITAs, and widely implemented for downloaded software (including updates) developed by a third party. *See, e.g.*, §VI.E above, (*citing* EX1034 (Cole), 304-306; EX1044 (Tanenbaum 2001), 641-642, 590-591, 643-644 (JDK 1.2); EX1045 (Garfinkel), 169-172, 265-266; EX1009 (Hall), 458)); *see also, e.g.*, §VIII.A.3 below; §IX.A below.

322. Vasell discloses development of the **service application software** in an “open” environment that included “different service providers” and “third party developers”, but does not explicitly describe an **application repository** storing **service applications** developed by these various different developers.

323. Vasell describes that the design of its “modular[]”, virtualized (*see* ¶304 above) **distributed system** “ensure the security, isolation and robustness of the **service applications**”, “achieve application integrity”, allow “multiple **service applications**... to simultaneously operate without detrimentally affecting each other”, and “prevent[] operation of a **service application** from disrupting other

service applications, thus ensuring integrity of the service gateway system.” *Id.*; EX1004 (Vasell Patent), 20:49-52, 18:12-13, 4:58-62, 5:11-15, 18:2-5, 12:47-54 (“Because of the need for a secure, robust environment for the service applications 70-72, integrity among the service applications 70-72 and between all applications and communications networks involving the service gateway system is desirable.”), 2:50-53 (“It is a further purpose of the service gateway system to allow multiple service gateway applications to simultaneously operate without detrimentally affecting each other.”), 3:19-24 (“While sharing the same infrastructure, the service applications can be modularized so that from the perspective of the service providers, a service gateway according to the present invention will appear as if it is dedicated to each service provider’s respective service.”).

324. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 3-4 (“Requirements of an e-service infrastructure” include that “The infrastructure must be flexible, open and modular, to accommodate a range of communication standards and protocols and to allow individual components rather than entire systems to be replaced as new technology is introduced.”); *see also* ¶304-306 above (*citing* EX1004 (Vasell Patent), 18:27-30; EX1016 (Vasell 2<sup>nd</sup> Provisional), 4, 7-9), ¶¶318-319 above (*citing* EX1004 (Vasell Patent), 11:64-12:10, 4:52-65, 15:46-47; EX1016 (Vasell 2<sup>nd</sup> Provisional), 3-4, 6-9).

325. Vasell discloses that the service gateway system is “remotely

managed”, for example, via “management system service 76” of “system services layer 110”, which “is “implemented in the form of the boxlets 64-69 executing within cells”, and that “provides an external interface through which the service applications ... may be downloaded, installed, removed, executed, and controlled”, which are management operations that Vasell describes, in detail, as being performed by “cell manager 93” using “cell table 94”:

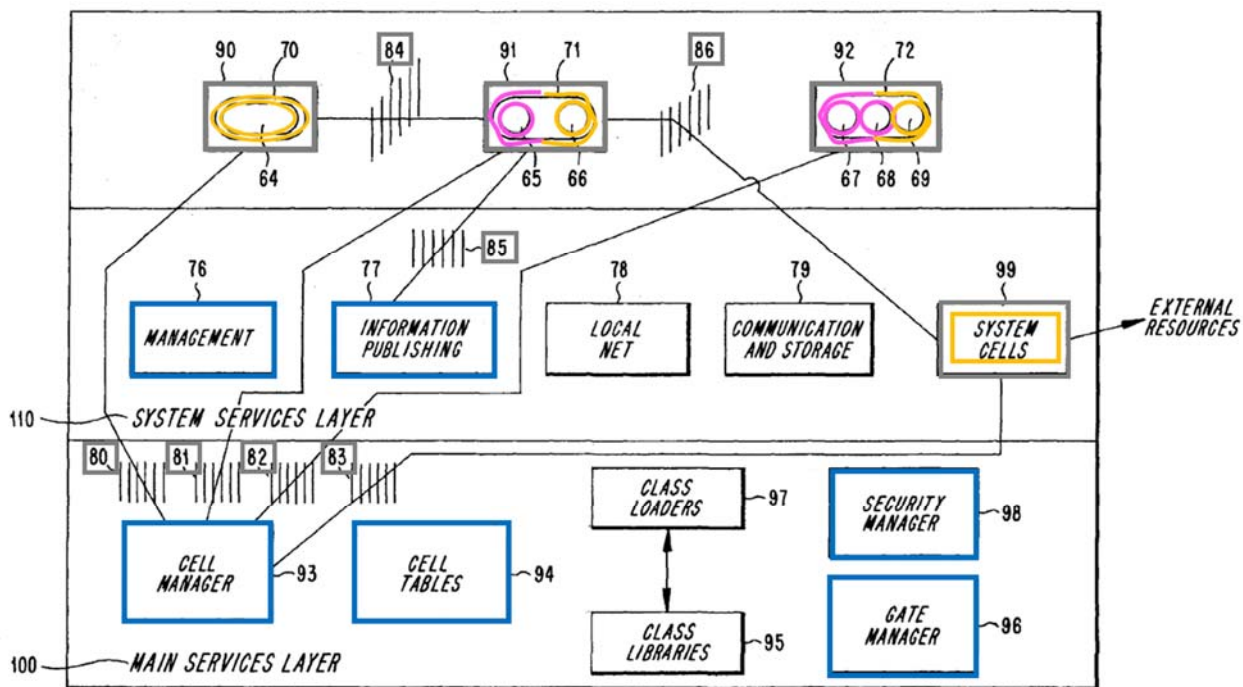


FIG. 6

EX1004 (Vasell Patent), FIG. 6 (which I have annotated above and refer to in my Declaration as “Annotated FIG. 6”), 12:11-29, 20:23-31, 22:34-37, 20:10-22:30, 15:58-17:39, 4:44-50, 5:16-26, 9:20-40.

326. Vasell 2<sup>nd</sup> Provisional describes this as follows. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“administer[ing] ... the infrastructure ... should [be] do[ne] ... using

a **management system** for operation and maintenance (O&M)”, “[a]t the architecture level, the network operator has a key role, by assuming the responsibility for maintenance and upgrades.”), 6 (“[i]t must be possible to manage an *e-box* remotely. This includes system software upgrades as well as supervision and error detection.”), Figure 4 (“Organization of the service platform”, including “**Information publishing (Web, WAP, etc.)**”, “**cell manager**”, “**cell table**”), 7 (“the *e-box* service platform ... permits the remote life-cycle management of **service applications**”), 8 (“The **management** of **service applications**—downloading, installation, execution, control, and removal—is performed via a specific **O&M interface** implemented as a system service boxlet. This system service is primarily used by the **management system**.”), 9 (description of “*management system (MS)*”, “*network operator (NO)*”).

327. Vasell discloses that the **management software** could also include, for example, “**diagnostic**” **software**, running on **network operator server 24** (and/or on **service provider equipment 34**) to “monitor, “maintain and control” distributed **service applications** running on a **service platform server(s)** and one or more **network operator** and/or **service provider server(s)** in the **service gateway system**. EX1004 (Vasell Patent), 9:20-41, 6:8-20, 3:2-5, 11:40-46 (*see* ¶307 above); EX1016 (Vasell 2<sup>nd</sup> Provisional), 6, 8 (“The system services layer (Figure 4) in the service platform provides several general services that a **service application** can make use of ... A system service is implemented exactly like any other service application; that is, in

the form of a boxlet executing in a cell. The interface offered by a system service is exported as a gate.”), 9 (Figure 5).

328. Vasell discloses that “system services layer 110” also includes “information publishing system service 77... through which the service applications ... can publish or transmit information for display ... to client terminals 20, 28 or 30 in both the local network 10 and external networks such as the Internet 26 of FIG. 2.” EX1004 (Vasell Patent), 22:32-34, 22:39-45, 20:23-31, Annotated FIG. 6; EX1016 (Vasell 2<sup>nd</sup> Provisional), Figure 4 (“Organization of the service platform”, including “Information publishing (Web, WAP, etc.)”).

329. Vasell describes several additional operations for the service gateway system that are facilitated through these remote management services. For example, Vasell identifies “upgrade software” as a remote management operation in which servers are “switched on the fly” to the upgraded “service application”, and describes an example of how the “cell manager 93” and “cell table 94” enable an “update to... a new configuration of one of the service applications”, even “if it is running”, such that the “service” provided by this one service application in the service gateway system “may continue undisturbed while [these] configuration updates take place” which “ensures smooth updates together with complete crash protection”:

The information stored in the cell table 94 allows the service platform server 22 to be switched on the fly to a new configuration of a service

**application** 70-72. For instance, the **cell manager 93** may be notified of a new configuration of one of the **service applications** 70-72 by a **service provider 34** of FIG. 2. The **cell manager 93** will then update to the new configuration by stopping the existing configuration of the **service application** 70-72, if it is running, and starting the new configuration. In this way, **service may continue undisturbed while configuration updates take place**. This ensures smooth updates together with complete crash protection.

EX1004 (Vasell Patent), 5:16-26, 22:19-30, 22:6-18, Annotated FIG. 6, 12:11-29, 2:56-59 (“It is a further purpose of the **service gateway system** that all configurations... be either locally or remotely downloadable.”).

330. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Service Provider Requirements of an E-Services Infrastructure: ... *Upgradability*: It must be easy to modify and upgrade services on the existing infrastructure, with a minimum of consumer involvement.”), 6 (“it must be possible to manage an *e-box* remotely. This includes system software upgrades as well as supervision and error detection.”).

331. As another example, Vasell discloses that “**cell manager 93** handles resource management so that no single one of the cells 90-92 consumes too much of the **service gateway system** ... resources, *e.g.*, CPU time, memory, persistent storage, and the like” and describes an example of how such system resources are “dynamically allocated” based on “predetermined” thresholds:

To achieve a secure, robust environment, different ones of the **service applications** 70-72 are not allowed to inadvertently interact in such a way that would cause them to fail or detrimentally affect each other. For example, one of the **service applications** 70-72 such as, e.g., a video telephone service application, may require a great deal of bandwidth for operation. Such a **service application** would not be allowed to entirely dominate the resources of a service gateway to the extent that other **service applications** are “starved” of the resources they need to run. Instead, resource requirements may be **dynamically allocated based upon a predetermined scheme of maximum allowed or minimum required resources, for a given situation.**

EX1004 (Vasell Patent), 12:54-63, 20:45-49, 14:15-17 (“Each of the cells 90-92 may be characterized according to its quota of resources (e.g., CPU, memory, persistent storage, and the like).”).

332. Vasell also describes the **service application** development environment of the **service gateway system** as an “open” environment that included “**different service providers**” and “**third party developers**”, identifies that “service gateway compatibility potentially opens **third-party markets** for... services”, and discloses that “**management system service 76** provides an **external interface** through which service applications ... may be downloaded, installed...”. EX1004 (Vasell Patent), 3:10-15, 5:27-33, 10:18-23, 13:37-43, 22:35-37.

333. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“*Development environment for **application software***: The development

environment must follow the ‘write-once, run-everywhere’ maxim and should be based on Java standards. New **applications** will interact with the **e-service infrastructure** through Java application program interfaces (API) that comply with mainstream Java development. By leveraging the Java development, the **application software** environment can be taken to a higher level of abstraction, allowing nonspecialists to develop **service applications** more easily.”), 8 (“*Development environment*: Boxlets are created using a standard Java development environment. For instance, the Java development kit (JDK) from Sun Microsystems can be used as well as other development environments. The only parts that are specific to boxlet development are the libraries that contain APIs for the main services and system services layers.”), 9 (identifying the disclosed “new **e-service infrastructure**” as “an open platform that can be used by several independent service providers”).

334. Although Vasell also identifies that “**service provider equipment 34** ... **may** be the initial repository for **software** used to implement connectivity based services” of such service provider, and that “**network operator server 24** may need to receive some portions of the **service application software** from the service provider” to then “download some or all of that [received] **software** to the **service platform server 22**”, Vasell does not explicitly disclose a **centralized application repository** storing the respective **service applications** developed by each of the “**different** service providers” and “**third party** developers” of its intended “open”

development environment. *Id.*; EX1004 (Vasell Patent), 7:14-18, 8:59-61, 8:29-36, 8:47-58.

## 2. The Alves Textbook (EX1008)

335. Alves (EX1008) is a textbook, titled “OSGi in Depth”, authored by A.d.C Alves, that the Manning Publications Company published in December 2011, was publicly accessible at least by March 2012, and was actually disseminated to interested persons exercising reasonable diligence at least by November 2012. EX1008 (Alves), 1, 4-5; EX1014 (Munford Librarian Declaration), ¶¶44-65.

336. Alves describes itself as providing an “*in-depth*” description of various technical implementation details of the OSGi open specifications (which are implemented in a Java run-time environment) for which Vasell (*see* §VIII.A.1 above) disclosed the foundational concepts (including implementation in a Java run-time environment) more than a decade earlier. EX1008 (Alves), xiii-xiv (“OSGi... has been around since the late 1990s...this book... looks in detail at many of the OSGi Core concepts while also elaborating on a number of vital technologies from the OSGi... Enterprise specifications.”), xviii (original emphasis); *see also* §VI.F.1 above (Technology Background – Distributed Systems – OSGi (¶¶177-190 above).

337. Alves states “the OSGi platform is an ideal platform for cloud computing.” EX1008, 250, 269, xiii-xiv, 261-263, 266-268.

338. Alves identifies that OSGi’s standardized Enterprise implementation

(EX1028, OSGi Service Platform Enterprise Specification) included communications between a locally-installed “bundle that’s exporting a remote service in one OSGi framework instance and [a remotely-installed] bundle that’s importing the service in another [OSGi] framework instance” using well-known protocols “such as RMI (Remote Method Invocation), SOAP (Simple Object Access Protocol), and REST (Representational State Transfer)”, which POSITAs understood to provide secure communications between software components of a distributed application:

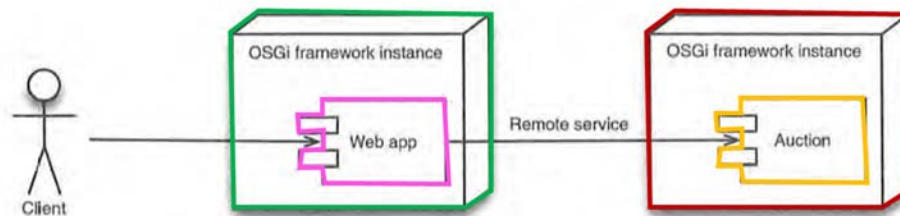


Figure 10.9 An auction system partitioned into two separate OSGi framework instances, communicating through an OSGi remote service

EX1008 (Alves), 249, 252-255, 261-262, Figure 10.9 (annotated above and showing an example of an “auction system” distributed application in which a “web app” bundle installed at a local machine securely communicates with an “auction” bundle installed at a “business tier” machine); see also §VI.F.1 above (Technology Background – Distributed Systems – OSGi ¶¶206-212 (identifying exemplary evidence of the well-known secure implementations of RMI, SOAP and REST) citing EX1046 (Oasis - SOAP Security Specification), 1-2, 7-8, 13-15; EX1047 (OWASP - WSS Security Specification), 1-2, 4; EX1048 (OWASP - REST Security

Specification), 1-2; EX1053 (JSSE Guide), 3-5, 68; EX1054 (Java RMI Overview), 1; EX1055 (Java RMI Socket Overview), 1; EX1056 (Java RMI SSL Overview), 1; *see also* §VI.F.1 above (Technology Background – Distributed Systems – OSGi), ¶205 above *citing* EX1008 (Alves), 261-263, Figure 10.10 (which I reproduced and annotated therein).

339. Alves discloses that, in a private/community/hybrid cloud OSGi implementation, where the network administrator (rather than a cloud provider) continues to “manage OSGi framework instances and their bundles” (*e.g.*, FIGS. 10.10 (*see* §VI.F.1 above), 10.12 (further below)), a “cloud provider could give [the network administrator] access to an OSGi bundle repository (OBR)” in the provider’s cloud, from which the administrator “could install, update, and uninstall bundles as needed”, and for which developers “can design **and validate** [their] applications using any OSGi implementation in a standalone environment before moving to the cloud.” EX1008 (Alves), 263, 266 (emphasis added in bold).

340. In this case, for any bundle updates, the software developer would “just need to inform the cloud provider of the update”, and “OSGi would let [the administrator] update [the] system quite easily and efficiently”, because “OSGi application[s] already hav[e] to deal with the fact that dynamic changes and updates may happen, so the fact that it’s happening when it’s running in the cloud or running as a standalone local application is mostly immaterial.” EX1008 (Alves), 266.

341. Moreover, Alves discloses the importance of software developers “validat[ing]” (testing and certifying as successfully completing such testing) their applications, e.g., before “moving [them] to the cloud” OBR, and technical implementation details for “effectively” and “eas[ily]” doing such application testing. For example, Alves discloses that application developers will “improve robustness by testing your applications”, states “[i]t’s a good idea to start by testing a bundle at the unit level ... to make sure that the bundle ... works in isolation ... The good news is that you can continue using JUnit to test your OSGi bundle ... because OSGi avoids imposing any of its technology-specific interfaces on the application ... [which] allows developers to test their classes as they normally would ...”, and describes technical implementation details for “effectively” conducting these “unit level” bundle tests. EX1008 (Alves), 126-128, 266.

342. Alves also states that, “[a]fter having unit tested all of [their] bundles, the next step” for application developers “is to make sure the bundles are able to talk to each other within the OSGi platform itself ... [using a] process ... called *integration testing*”, and describes technical implementation details for “effectively” conducting this “integration testing.” EX1008 (Alves), 128-129 (italicized emphasis in the original); *see also* Alves at 130 (“The OSGi framework imposes very few vendor interfaces into the application classes, so **it’s generally easy** to unit test a bundle by treating it as a regular Java class. Spring DM provides several mock

classes for the OSGi framework API. Spring DM also allows you to perform integration tests. It has helper classes that start an OSGi platform, install **bundles**, and allow you to test their integration.”).

343. Accordingly, Alves states that another “important advantage” of this private/community/hybrid OSGi cloud implementation is that the software developer “can design and validate [its] **applications** using any OSGi implementation in a standalone environment before moving” the validated (tested and certified as successfully tested) bundles to the cloud **OBR**. *Id.*

344. Alves provides a further example in which the private/community/hybrid cloud OSGi Enterprise implementation (*e.g.*, FIG. 10.10, (reproduced and annotated in §VI.F.1 above)) utilizes the separate cloud provider’s **OBR** (FIG. 10.12 (reproduced and annotated below)):

We have several OSGi **bundles** that have been uploaded to the **cloud’s OBR**. In addition, we’ve provisioned OSGi framework instances to computing nodes, as shown in figure 10.12. For example, we have two OSGi framework instances in the **client tier**; therefore, we could assign each to a different **computing node**. Next, we have the remaining eight OSGi instances, which we could likewise assign to a different **computing node**. Finally, we could install the **web application [bundles]** in the OSGi instances of **the client tier** and the **auction application bundles** in the OSGi instances of **the business tier**. And, of course, we would need to make sure that the **web application bundle**

could invoke remote services residing in the **bundles** of the **business tier**. In this environment, we can easily **manage** updates to our system by leveraging the OSGi framework.

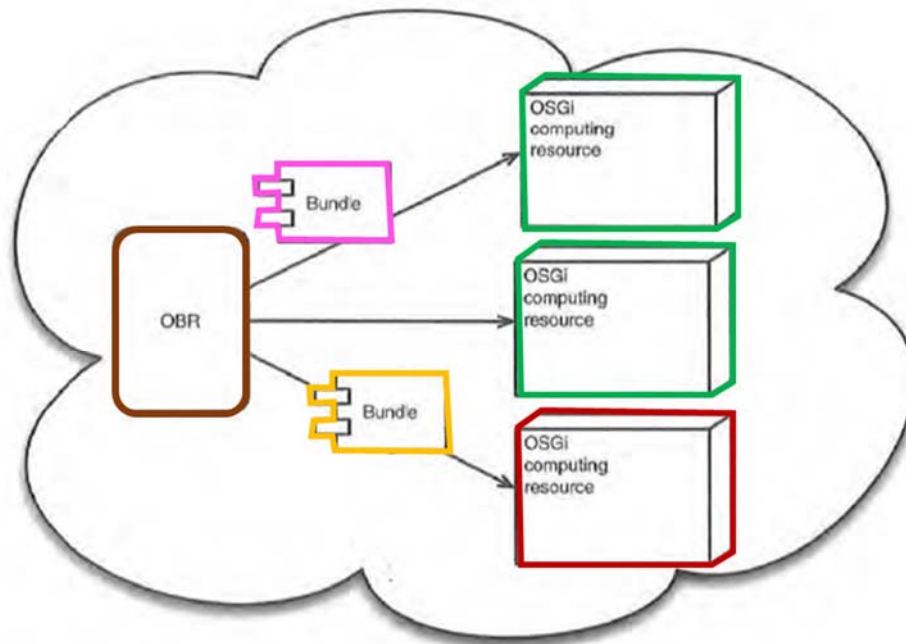


Figure 10.12 OBR and OSGi framework as a platform for cloud computing

EX1008 (Alves), 263, 266-267.

345. Alves also discloses that, in a public cloud OSGi Enterprise implementation, where a cloud provider (and not the network administrator) “**manages** OSGi framework instances and their **bundles**” that are “hosted in **machines** managed by the cloud provider” (e.g., EX1008 (Alves), FIGS. 10.11, 10.13), well-known “elasticity in the **cloud**” (what the patent calls “cloud breathing”) is supported by the OSGi platform. EX1008 (Alves), 266-268 (“[W]e upload the **bundles** to the **OBR** in the [provider’s] cloud, and each **bundle** tells the [provider’s] cloud its requirements... The [provider’s] cloud infrastructure would consider all

these requirements and assign the **bundles** to the correct OSGi framework instances.”); *see also* §VI.F.2 above (EX1012 (Kachele 2013), 19-22).

### 3. The Hall Textbook (EX1009)

346. Hall (EX1009) is a textbook, titled “OSGi in Action - Creating Modular Applications in Java”, authored by Richard S. Hall, et al., that the Manning Publications Company published in April 2011, was publicly accessible no later than December 2011, and was actually disseminated to interested persons exercising reasonable diligence at least by December 2011. EX1009 (Hall), 1, 4-5; EX1014 (Munford Librarian Declaration), ¶¶66-87.

347. Hall describes itself as a textbook that “thoroughly describe[s] the important aspects of OSGi and show[s] how to use them” in the in the standardized Java run-time environment. EX1009 (Hall), xx.

348. Hall identifies that “the OSGi framework is designed to take advantage” of “the built-in support for secure sandboxes” in “the Java platform”, including assigning permissions to “**bundles**”—called “boxlets” or distributed “**service application** portions” in Vasell (*see* §VIII.A.1 above)—and having the AccessController and SecurityManager components of the Java virtual machine (JVM) allow or deny access to **bundles** based on such permissions. EX1009 (Hall), 438, 440-448, 464; *see also* my Technology Overview §VI.D.3 above (Java Virtual Machines and Java Applications) (*citing* EX1034 (Cole), 304-305, 406; EX1045

(Garfinkel), 44-45).

349. Hall states that “OSGi uses ... the Java security model ... to provide the infrastructure to deploy and manage applications that must run in secure environments”, and that “Java security provides the foundation of the OSGi security model.” EX1009 (Hall), 475.

350. Hall explains that additional well-known Java security features that “can be used for OSGi”, especially in environments where **bundles** (and associated upgrades) are developed by third parties, are digital signatures and associated certificate (issued by a “well-known (trusted)” party). EX1009, 457-463, 331-334, 472-476; *see also* Technology Overview §VI.E above (Security for Software Implemented and Downloaded Over the Internet Including Digital Signature Implementations) (*citing* EX1050 (OSGi 2000 Overview), 8-10; EX1067 (DSS FIPS PUB); EX1034 (Cole), 304-306; EX1044 (Tanenbaum 2001), 641-642; EX1045 (Garfinkel), 170-172); Technology Overview §VI.B.1 above (Hash Functions and Their Uses) (*citing* EX1057 (Microsoft Computer Dictionary), 432, 131, 145, 388-389).

351. Specifically, Hall explains that “the jarsigner tool from the standard Java SDK” “can be used for OSGi”, whereby each developer may be required to “digitally sign” its respectively developed **bundles** (and associated upgrades) using the unique private key issued to it by a “well-known (trusted)” party, and whereby

the “management agent” (and each “JVM security manager” of the JVMs in which the bundles are deployed) of the OSGi environment would use the corresponding unique public key (identified in the X.509 certificate of the developer, signed by the “well-known (trusted)” party, and “mathematically related” to the developer-specific private key) to “authenticate the provider of [the] bundle” (by “verify[ing] that the signer has access to the private key” (generated by the “well-known (trusted)” party identified in the X.509 certificate) and “ensure that the bundle content has not been modified.” EX1009 (Hall), 457-463, 331-334, 471-476.

352. Hall states, “[i]n OSGi, the signer of a bundle is associated with it” and, “[w]ith this association,” the management agent of the OSGi environment would “grant permissions to a bundle based on its signers”, *i.e.*, based on the identity of the bundle developer. *Id.*; EX1009 (Hall), 459, 476 (“You can make your life a lot simpler by signing your bundles with certificates and assigning permissions to bundles based on who signed them.”), 458 (“Digital cryptography terminology ... The public key is shared with others in the form of a certificate, which they can use to verify that a signature was generated with the private key. ... Certificate: A form of metadata about a public key, binding it to the identity of the private key holder. This binding is achieved by having a well-known (trusted) third party sign the public key/identity pair.”).

353. Hall also explains that this same well-known “digital signing”, and

authenticity and integrity verification, was applicable to both original, and updated, versions of **bundles** to render them “tamperproof.” *Id.*; EX1009 (Hall), 331-333, 179 (“in general it’s better to update the whole **bundle**”); *see also* Technology Overview §VI.E above (Security for Software Implemented and Downloaded Over the Internet Including Digital Signature Implementations) (*citing* EX1044 (Tanenbaum 2001), 590-591, 643-644 (JDK 1.2); EX1045 (Garfinkel), 169-172, 265-266; EX1050 (OSGi 2000 Overview), 8-10).

354. Additionally, Hall explains that the “JVM security manager”, of the JVM(s) in which the bundles are respectively deployed (*see, e.g.*, §VI.F.1 above, ¶192 above)), performs digital signature matching to determine these bundle-specific “permissions.” EX1009 (Hall), 471 “[T]he [OSGi] framework sets the JVM security manager to an implementation-specific [security property value] when started.”), 439-440 (“OSGi uses ... the Java security model ... to securely deploy and manage applications ... Providing meaningful security management involves ... [e]stablishing identity ... [and] [p]erforming permission checks and privilege management ... identity can be established by ... cryptographic measures using digital certificates ... [l]ast but not least, security must [include] ... internal security checks to prevent external code from performing undesired operations and also how to limit privileges so external code can perform potentially sensitive operations in a safe way ... [T]he OSGi security model is based” on “the Java security architecture

and its permission model...”), 441-448 (“The OSGi framework security model relies on Java’s domain-based approach ... in OSGi ... a [protection] domain is mapped one-to-one to a bundle, you can think of it as a bundle protection domain ... To understand how protection domains enable permission checking, consider code that performs a sensitive operation ... The code in the JRE [Java Runtime Environment] ... performs security checks internally to make sure the invoking code has permission to perform the operation ... [t]he JVM checks permissions by collecting all protection domains associated with classes on the call stack and seeing if each involved protection domain has the specified permission granted to it ... [T]he actual permission granted [to a bundle] is controlled by its constructor parameters ... [whereby] an LDAP filter ... matches against ... signer—Identity information about the bundle provider ...”); *see also id.* at 459-463. As discussed in §VI.F.1 above (¶192 above), each JVM, in the OSGi framework environment, is running on a host OS of a network device (*e.g.*, router, gateway, server, etc.). *Id.*

355. Based on the determined bundle-specific “permissions”, Hall explains that this “JVM security manager” then “control[s] [each bundle’s] access to sensitive framework operations and information”, “control[s] which services a bundle can provide or use”, makes bundle-specific “deny access decisions”, determines the local resources “required by the bundle”, “enforces” local resource restrictions “at execution time”, etc. EX1009 (Hall), 439-440, 444-447, 464, 472-473.

356. Moreover, related to the well-known “update” operation of OSGi’s lifecycle management feature, Hall confirms “[o]ne of the strengths of OSGi is that you’re able to install, update, and uninstall **bundles** without having to restart the JVM.” EX1009 (Hall), 283.

#### 4. Rellermeyer (EX1011)

357. Rellermeyer (EX1011) is an IEEE article, titled “Dependability as a cloud service - a modular approach,” authored by J. Rellermeyer et al., that was presented at the 2012 IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012), Boston, MA, USA, 2012, was published in the conference proceedings by IEEE in August 2012, was publicly accessible no later than late-September 2012, and was actually disseminated to interested persons exercising reasonable diligence at least by April 2013. EX1014 (Munford Librarian Declaration), ¶¶20-43.

358. Rellermeyer discloses a “service ... call[ed] **CLOUDDP**... designed to turn OSGi services”—“objects that are specifically shared between [**application**] **components**” called “*bundles*”—into elastic units of deployment which can be migrated and replicated across multiple **cloud nodes** to increase the availability and reliability of the administered service.” EX1011 (Rellermeyer), 2-4.

359. Rellermeyer discloses that the **CLOUDDP** service “can be architected using features **already provided** by OSGi” because, for example, (i) “the

compositional approach of modular software... is key to building scalable and dependable systems across a varying set of **machines in the cloud**", (ii) OSGi's use of "services... as a design element... result[s]... [in] less entangled and more flexible" systems, and (iii) the "OSGi Enterprise Specifications" already detail "the **API of the remote service admin**" ("the **interfaces through which a possible topology manager can interact**") and "the mechanism for importing **remote** services into... a **local** OSGI framework and exporting **local** OSGi services" to the **remote** service by creating "a **local** proxy for the **remote** service". EX1011 (Rellermeyer), 2-4.

360. Rellermeyer discloses "**CLOUDDEP**... takes over the role of a... local topology manager on each node of the cloud deployment" and "**monitors** the **[remote]** service invocations" on **local** ("**client**") **nodes**, and explains "load-balancing" (Figure 2 (below, annotated), "case 1") and "elastic scalability" (*id.*, "case 3") usage cases, which provide the identical "cloud breathing" functionality disclosed in the '823 Patent (*see* §VII.A above):

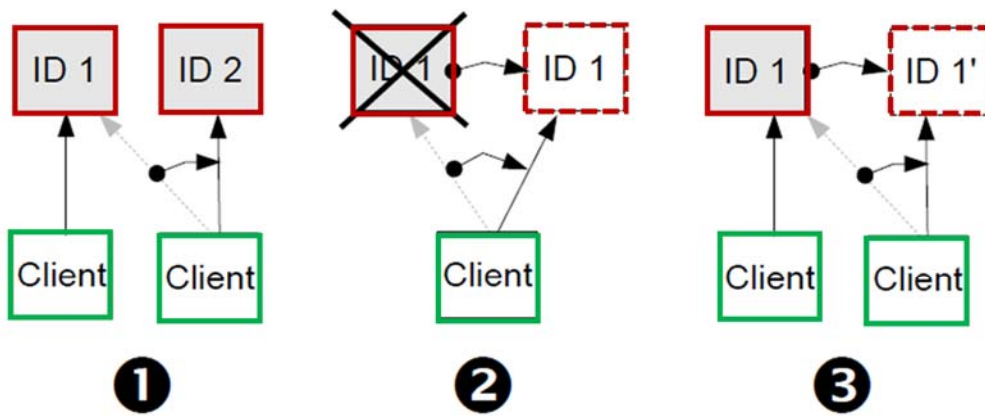


Figure 2. Different Use Cases for a Dependability Service. 1: Load balancing, 2: Failover, 3: Elastic Scalability.

EX1011 (Rellermeyer), 4-5.

361. In Case 1 for “load balancing”, “**CLOUDDEP**... instrument[s] the **local** proxy to gather information about the frequencies of [**remote**] service invocation and response times”, and “[i]n the event that one **remote** service **instance** [“ID1”] **becomes overloaded**, detectable by an increase of invocations and an increase of the response time, **CLOUDDEP** can re-bind a **client** to a different [**remote**] service **instance** [“ID2”], thereby re-distributing the load among the [**remote**] service **instances.**” EX1011 (Rellermeyer), 4-5.

362. In Case 3 for “elastic scalability”, “[i]n the event of a [**remote**] service **over-utilization**, **CLOUDDEP** can start a new **instance** of the [**remote**] service [“ID1’”] and bind some of the **clients** to the new service” and “[s]cal[es] back to a smaller number of [**remote**] service **instances.**... by... moving all **clients** of a [**remote**] service to a different existing **instance** [“ID1”] and then dropping the service **instance**

[“ID1”].” EX1011 (Rellermeyer), 5.

## **B. Claim Construction**

363. I have been informed by counsel that, in order to properly evaluate the Challenged Claims of the '823 Patent, the terms of the claims must first be interpreted. As I discussed in §IV.A above (Understanding of Relevant Legal Principles – Claim Construction) above, I am informed by counsel and understand that the terms in the Challenged Claims of the '823 Patent are to be given their plain and ordinary meaning as would have been understood by a person of ordinary skill in the art (“POSITA”, defined in §V above) as of the Priority Date (*see* §VII.C above) in view of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. That is the meaning I have applied in my analysis.

364. I have been informed by counsel that no court or administrative board has construed any of the claims of the '823 Patent. If a board or court construes any term of the '823 Patent, I reserve the right to supplement my opinions in view of such construction.

365. I have been informed by counsel that the Petitioner and the Patent Owner, in the currently pending District Court litigation between them, have proposed competing constructions of several claim terms in the '823 Patent, and in the related '095 Patent and '823 Patent. I have reviewed these proposed

constructions for these claim terms, including in the claim construction briefs filed in this litigation. *See* EX1070 (Patent Owner's Opening Claim Construction Brief); EX1071 (Defendant Microsoft Corporation's Responsive Claim Construction Brief); EX1071 (Petitioner's Responsive Claim Construction Brief). I note that neither Petitioner nor Patent Owner contends that the meaning of any claim term deviates from its plain and ordinary meaning due to any lexicography, disclaimer or disavowal in the intrinsic evidence. *Id.*

366. I have also been informed by counsel that the Board does not construe claim terms unless the terms are in controversy, and then only to the extent necessary to resolve the controversy for purposes of the *inter partes* review proceeding.

367. I have followed these principles in my analysis throughout this Declaration. It is my opinion that no specific construction of any term is required because the relied-upon prior art meets each of the claimed terms under any proper construction (including under both the Petitioner's and the Patent Owner's proposed constructions of the claim terms in the District Court litigation).

368. I have also been asked to consider, in the alternative, that certain claim elements (which I identify below) may be subject to interpretation as a means for performing a recited function as set out in 35 U.S.C. § 112(f). I have not been asked to opine as to which claim elements, if any, should be subject to such interpretation. I have also not been asked to opine as to what, if anything, within the specification

of the '871 Patent may constitute sufficient disclosure of any structure linked to performing the recited functions for such claim elements.

369. I am informed by counsel that the Patent Owner may argue, and/or the Board may find, that the below-identified claim elements of the Challenged Claims are subject to interpretation as a means for performing a recited function as set out in 35 U.S.C. § 112(f). Accordingly, I have also considered in my analyses herein, in the alternative, each of the below-identified claim elements as subject to interpretation as a means for performing a recited function as set out in 35 U.S.C. § 112(f). I have further considered the specific portions of the specification that the Patent Owner identifies in Patent Owner's Opening Claim Construction Brief (EX1070) in the Related Litigation as describing the structure corresponding to the recited function for each such claim element, and any additional portions of the specification identified by Petitioner in its Responsive Claim Construction Brief (EX1071) in the Related Litigation.

370. In the event that one or more of these constructions is changed, or in the event that additional terms not specifically construed herein receive a proposed construction, I reserve the right to revisit my analysis under such additional construction(s).

**1. "virtual fabric" (in claim 12):**

371. This claim term is recited in claim 12 (§IX.B.3 below), which recites:

“a **virtual fabric** which provides a secure communication layer for said at least one of the plurality of network device applications and said at least one of the plurality of cloud applications.”

372. Patent Owner identifies “an abstraction layer for applications to communicate with each other”, as described in EX1001 (’823 Patent), 13:33-39, 20:36-41, 11:57-62 (which is a shared specification with the ’871 Patent and the ’095 Patent), as the structure corresponding to the recited function and the portions of the specification describing this structure. EX1070 (Patent Owner Claim Construction Brief), 9-10; EX1071 (Petitioner Claim Construction Brief), 12-13.

**2. “load controller adapted to monitor loads ... and affect change in accordance with thresholds received ...” (claim 7)**

373. This claim term is recited in claim 7 (§IX.C.4 below), which recites: “wherein the DRS further includes a **load controller adapted to monitor loads** on at least one of the plurality of network device applications and at least one of the plurality of cloud applications **and effect change in accordance with thresholds received** from the application management portal.”

374. Patent Owner identifies “a controller executing at least one of a ‘cloud breathing’ or ‘load monitoring’ application”, as described at EX1001 (’823 Patent), 21:66-22:5, 22:20-22 (load monitoring application), 15:64, 22:9-26, FIGS. 17A-17B (cloud breathing application), which is a shared specification with the ’871 Patent and the ’095 Patent, as the structure corresponding to the recited function and the

portions of the specification describing this structure. EX1070 (Patent Owner Claim Construction Brief), 16-19; EX1071 (Petitioner Claim Construction Brief), 19-20.

**IX. CHALLENGES: CLAIMS 1-2, 12-15 AND 19 OF THE '823 PATENT ARE OBVIOUS OVER VASELL IN VIEW OF ALVES AND RELLERMEYER, AND CLAIMS 3-5, 7-8 AND 18 OF THE '823 PATENT ARE OBVIOUS OVER VASELL IN VIEW OF ALVES, RELLERMEYER AND HALL**

**A. Motivation to Combine the Teachings of the Prior Art References**

375. As I discussed in my Technology Background section on Distributed Systems - Open Standards Gateway Initiative (OSGi) (§VI.F.1 above, ¶¶177-190 above), in my Overview of the Prior Art in the Grounds for Challenge sections (§§VIII.A.1-VIII.A.4 above), in this section, and in my claim-element-by-claim-element detailed analysis section (§IX.B below), Vasell (*see* §VIII.A.1 above) unsurprisingly leaves certain implementation details of its **service gateway system** to design choices of POSITAs. However, as I discussed, for example, in my Technology Background section on Distributed Systems - Open Standards Gateway Initiative (OSGi) (*see* §VI.F.1 above, ¶¶177-190 above), Vasell's disclosed architecture and methods (including implementation in a Java run-time environment) were subsequently incorporated into the Open Standards Gateway Initiative (OSGi) specifications. Technical implementation details that were intended to implement Vasell's teachings are described in each of the published references cited in Grounds 1 and 2 -- Alves (EX1008, *see* §VIII.A.2 above),

Rellermeyer (EX1011, *see* §VIII.A.4 above) and Hall (EX1009, *see* §VIII.A.3 above).

376. Accordingly, the motivation for POSITAs to modify the teachings of Vasell, to include the features described in the published references for their intended purpose, and to achieve predictable benefits with a reasonable expectation of success, is demonstrated by the references themselves.

377. For example, Vasell discloses a **service gateway system** in which connectivity-based services are implemented using **service applications** whose processing is **distributed** across respective software components, executing in respective JVMs hosted on respective OSs of **local** and **remote** servers and securely communicating with each other, where a network operator administrator remotely, via an **external management interface**, and **management software modules** also executing on the **service gateway system servers**, **manages** the downloading, installing, upgrading and termination of these distributed **service application** components, and **monitors** and **controls** their implementation on the **service gateway system servers** (including dynamic device resource allocation), and for which many different service providers and third parties respectively develop the distributed **service applications**, and their underlying software components, for the many different connectivity-based services. *See, e.g.*, §VIII.A.1 above (Overview of the Prior art in the Grounds for Challenge - Vasell).

378. A POSITA would have been motivated to modify Vasell’s teachings with a **centralized application repository** for storing distributed **applications** for which developers can design and validate their **applications** using any OSGi implementation in a standalone environment before moving to the cloud **repository** as taught by Alves because it provides the benefit of increased flexibility and security. EX1008 (Alves), 263, 266.

379. Specifically, Alves describes technical details for implementing OSGi-compatible **distributed systems** in various well-known cloud environments (including for utilizing a cloud provider’s **OSGi bundle repository (OBR)** for storing “validate[d]” (tested and certified as successfully tested) **bundles** (OSGi software components of distributed **applications**) for subsequent installation in **local** and **remote** servers in an OSGi-compatible **distributed system**), and technical implementation details for using standardized well-known protocols to provide secure communications between **locally** and **remotely** installed software components of distributed **applications** implementing services. *See, e.g.*, §VIII.A.2 above (Overview of the Prior art in the Grounds for Challenge – Alves) (*citing* EX1008 (Alves), xiii-xiv, xviii, 263, 266-267, FIGS. 10.10, 10.12).

380. A POSITA would have been motivated to modify Vasell’s teachings with **management software** that provides for load balancing, failover, and elastic scalability as taught by Rellermeier. Specifically, Rellermeier discloses a

management service that, “using features already provided by OSGi” specifications, has the OSGi management software monitoring and controlling automatic load (resource usage) expansion/contraction across multiple cloud servers. See, e.g., §VIII.A.4 above (Overview of the Prior art in the Grounds for Challenge – Rellermeyer) (citing EX1011 (Rellermeyer), 2-5, Figure 2).

381. A POSITA would have been motivated to modify Vasell’s teaching with management software, and JVM security manager, implementations that require digital signatures and associated certificates for distributed applications as taught by Hall to provide increased security, and certification of the applications’ authenticity and integrity by a “well-known (trusted)” party, for the use of third party applications. Specifically, Hall discloses technical implementation details for having the OSGi management software, and each “JVM security manager” of the JVMs, each of which is running on a host OS of a network device in which the bundles are respectively deployed, implement well-known and standardized digital signature, certificate and key exchange techniques to verify the respective authenticity and integrity of each downloaded software bundle (and of each updated bundle version) as certified by the “well-known (trusted)” party. See, e.g., §VIII.A.3 above (Overview of the Prior Art in the Grounds for Challenge – Hall) (citing EX1009 (Hall), 179, 331-334, 457-463, 472-476).

382. POSITAs understood that the beneficial aspects that Alves,

Rellermeyer and Hall (§§VIII.A.2-VIII.A.4 above) described for OSGi **distributed systems**—increased scalability, having a centralized **repository** for storing “validate[d]” (tested and certified as successfully tested) distributed **software application** components (and associated updates) developed by different application developers (of varying trustworthiness), increased security, and being able to grant **application-specific-permissions** based on the respective **application** developer’s identity (as certified by a “well-known (trusted)” party), and being able to elastically deploy cloud resources to increase the availability and reliability of services—would be identically beneficial to Vasell’s foundational **service gateway system**. See §§VIII.A.2-VIII.A.4 above; §VIII.A.1 above (Overview of the Prior art in the Grounds for Challenge - Vasell); *see also* Technology Background section on Distributed Systems - Open Standards Gateway Initiative (OSGi) (§VI.F.1 above, ¶¶177-190 above).

383. POSITAs would have been motivated to modify Vasell’s teachings to (1) utilize a cloud-provider **application repository** of “validate[d]” (tested and certified as successfully tested) **applications** in a cloud environment, (2) implement inter-application communication protocols with well-known secure communication capabilities, (3) implement well-known and standardized digital signature, certificate and key exchange techniques, and (4) to have its **remote manager** monitor and control load expansion/contraction across multiple **cloud servers**, in order to

achieve the known and predictable benefits associated therewith.

384. In view of this Motivation to Combine section (and as I discuss further below in my claim-element-by-claim-element detailed analysis §§IX.B-IX.C below), I have demonstrated that a POSITA would have understood at least the following **seven** motivations to combine these teachings of these prior art references.

385. **First**, Vasell (§VIII.A.1 above) and Alves (§VIII.A.2 above) themselves provided the motivation that would have led POSITAs to implement Vasell’s network of **service provider servers**, and **network operator server(s)**, as a “cloud” using a **cloud-provider-application repository**, so that “many different types of services” could be implemented using “validate[d]” (tested and certified as successfully tested) software components of distributed **applications** developed in an “open” environment that included “different service providers” and “third party developers” and “open[ed] third-party markets for... services.” See §IX.B.1 below (claim elements [1.2]-[1.3]); §VIII.A.1 above (Vasell); §VIII.A.2 above (Alves).

386. **Second**, the general knowledge in the art was that various shared features of Enterprise OSGi and Vasell – listed below – made OSGi (and Vasell) an “ideal” system to implement in a well-known “cloud” environment, including where a private/community/hybrid cloud implementation would continue to be **managed** by the network operator as disclosed by Vasell, and to utilize a cloud-provider **application repository/store** to store “validate[d]” (tested and certified as

successfully tested) distributed **service application** components for respective installation in respective **servers** in such a **distributed system**. See §IX.B.1 below (claim element [1.2]); §IX.B.7 below (claim element [19.6] below); §IX.C.1 below (claim 3); *see also* §VI.F.1 above (OSGi); §VIII.A.2 above (Alves). These Enterprise OSGi and Vasell shared features included the following four attributes:

387. (A) software modularity (*see* §IX.B.1 below (claim elements [1.0]-[1.3] below); §VIII.A.1 above (Vasell); §VI.F.1 above (OSGi));

388. (B) distributed execution of **service applications** in a two-tier, multiple machine (**local network device** and **remote (cloud) server**), virtualized (JVM) architecture (*see* §IX.B.1 below (claim elements [1.0]-[1.3] below); §VIII.A.1 above (Vasell); §VI.F.1 above (OSGi); §VIII.A.2 above (Alves); §VIII.A.3 above (Hall)).

389. (C) dynamic **service application** component updates without affecting the concurrent execution of other **service applications** on the same **servers** (*see* §IX.C.6 below (claim 18); §VIII.A.1 above (Vasell); §VI.F.1 above (OSGi); §VIII.A.2 above (Alves); §VIII.A.3 above (Hall)); and

390. (D) dynamic resource usage (load) monitoring and control (*see* §IX.B.5 below (claim 14); §IX.C.3 below (claim 5); §IX.C.4 below (claim 7); §VIII.A.1 above (Vasell); §VIII.A.2 above (Alves); §VIII.A.4 above (Rellermeyer)).

391. **Third**, the beneficial aspects of having, in Vasell's foundational system, a centralized **repository** for **software** (and **software** updates) developed by

different application developers, and requiring each such developer to “design and validate” its **software** (and updates) in its own “standalone environment before moving” such **software** to the **cloud-provider-repository**, as described in Alves, would be accomplished as a mere application of a known centralized repository solution to a known **distributed system** ready for improvement to achieve known and predictable results. *See* §IX.B.1 below (claim element [1.2]); §IX.B.7 below (claim element [19.6] below); §IX.C.1 below (claim 3).

392. **Fourth**, Vasell (§VIII.A.1 above) and Rellermeyer (§VIII.A.4 above) themselves provided the motivation that would have led POSITAs to improve, “using features already provided by OSGi” specifications, Vasell’s **remote manager** monitoring and dynamically allocating **service gateway system** resources (*e.g.*, “CPU time, memory, persistent storage, and the like”) using Rellermeyer’s teachings of the OSGi **management software** monitoring and controlling load expansion/contraction across multiple **cloud servers**. *See* §IX.B.1 below (claim element [1.2]); §IX.B.5 below (claim 14); §IX.C.3 below (claim 5); §IX.C.4 below (claim 7); §VIII.A.1 above (Vasell); §VIII.A.4 above (Rellermeyer).

393. **Fifth**, the beneficial aspects of this combination of these teachings of Vasell with these teachings of Rellermeyer, including turning Vasell’s connectivity-based services “into elastic units of deployment which can be migrated and replicated across multiple **cloud nodes** to increase the availability and reliability of

the administered service”, would be accomplished “using features already provided by OSGi” specifications, and with the knowledge that OSGi’s and Vasell’s shared features made OSGi (and Vasell) an “ideal” system to implement in a well-known “cloud” environment, would give POSITAs confidence that the combination would have a high likelihood of success. *Id.*; *see also* §IX.B.1 below (claim element [1.2]); §IX.B.5 below (claim 14); §IX.C.3 below (claim 5); §IX.C.4 below (claim 7).

394. **Sixth**, Vasell (§VIII.A.1 above) and Hall (§VIII.A.3 above) themselves provided the motivation that would have led POSITAs to implement Vasell’s administrator, and **management software**, managed upgrades of software components of distributed **service applications**, in Vasell’s “open” development environment including “different service providers” and “third party developers” (of varying trustworthiness), using well-known and standardized “authentication”, “authorization”, and “identity verification” techniques— namely, Hall’s teachings of implementing well-known digital signature, certificate and key exchange techniques in OSGi’s JVM environment (shared with Vasell) — to respectively verify the authenticity and integrity of each such software component as certified by a “well-known (trusted)” party. *See* §IX.C.6 below (claim 18); §VIII.A.1 above (Vasell); §VIII.A.3 above (Hall).

395. **Seventh**, the beneficial aspects of this combination of these teachings of Vasell with these teachings of Hall, including rendering each distributed

application “portion”/component (and updates) “tamperproof”, authenticating the provider of such updates (by digital signature matching, which verifies the signer used the unique private key issued by the “well-known (trusted)” party identified on the X.509 certificate (with the unique public key of the key pair)), ensuring that the content of such update is unmodified, and allowing Vasell’s management services to grant Vasell’s application-cell-specific-access-levels to each such distributed application “portion”/component (and updates) based on the respective developer’s identity (signature, as certified by the “well-known (trusted)” party), have the JVM(s) running on the host OS of the recipient device of each such distributed application “portion”/component (and updates) perform standard Java digital signature and certificate matching to determine such application-cell-specific-access-levels and control the distributed application “portion[’s]” (and updates’) access to operations, information, services and device resources, would be accomplished using well-known and standardized security techniques in OSGi’s and Vasell’s shared centralized management and virtualized (JVM) architecture, as detailed by Hall, and, accordingly, POSITAs would have been confident that the combination would have a high likelihood of success. *Id.*; *see also* §VIII.A.1 above (Vasell); §VIII.A.3 above (Hall).

**B. Claim Element by Claim Element Invalidity Analysis: '823 Patent Claims 1-2, 12-15, and 19 are Obvious Over Vasell in View of Alves and Rellermeyer.**

396. As I identify in ¶61 above, throughout my Declaration, I use the color scheme that I identify in the table included in ¶61 above to annotate the same concepts in the '823 Patent (EX1001) and the prior art references that I discuss herein.

**1. Claim 1**

***[1.0] A system comprising:***

397. To the extent the preamble is limiting, Vasell discloses this limitation because it discloses a “**service gateway system**” that utilizes “**distributed processing**” across “servers or processors” connected to each other over “an Internet Protocol (IP) network such as the Internet” (*a system*).

398. One “illustrative” example of the “**service gateway system**” is “the entire system shown in FIG. 2”, which shows, as I have annotated below (*see my explanation of my annotations in ¶293 above and ¶403 below*) and refer to throughout my Declaration as “Annotated FIG. 2”, a “**service platform server 22**” “connected to the local area network 10” and “connected to [a] **network operator server 24** via the Internet”, which is connected to a plurality of “**service provider equipment 34** via... Internet 26”:

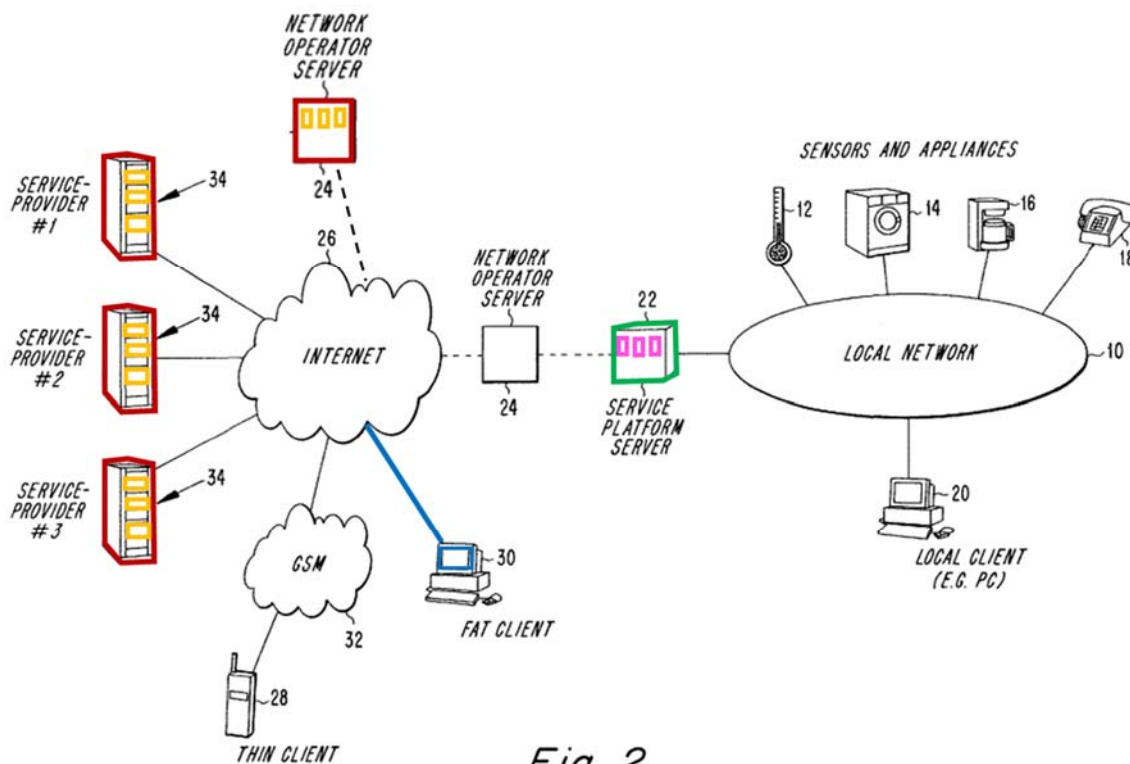


Fig. 2

EX1004 (Vasell Patent), Annotated FIG. 2, 4:31-47, 5:61-6:20, 6:62-7:14; EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Services should be implemented as distributed applications ...”), 5 (Figure 2), 9 (Figure 5).

399. Vasell discloses a “service gateway system” that “provides a platform for connectivity-based services”, where “[e]ach service may comprise a set of functionalities and logic which are implemented as software service applications”, and where each “service application may be distributed among various pieces of equipment which are geographically separated.” EX1004 (Vasell Patent), 4:31-50, 2:64-3:6, 1:41-45, 8:53-9:18 (see text reproduced in my analysis of claim element [1.3] below in §IX.B.1 above), 12:21-33 (“the software elements of the service

gateway system may be distributed among various units of geographically separated equipment, as shown in FIG. 2”), 11:13-58 (see text reproduced in my analysis of claim elements [1.2]-[1.3] below in §IX.B.1 above), Annotated FIG. 2 (above), Annotated FIG. 5 (see my analysis of claim element [1.1] below in §IX.B.1 above).

400. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Services should be implemented as **distributed applications**. By executing over several infrastructure nodes, more complex services can be developed without the need for extremely powerful and complex **edge servers**. Distributed **applications** will considerably increase the technical life and reduce the cost of the **edge servers**.”), Figure 2 (“The Ericsson **e-service infrastructure**” (which I have annotated (similar to Annotated FIG. 2 above) and reproduced below), 7 (“A **service application** typically fulfills the role of a gateway between **servers** in the external network (from which the service is accessed and controlled) and **devices** in the local network.”), 8 (description of “**service applications**” and “system services”), 9 (Figure 5, “Box D, Distributed Services System Architecture” including, e.g., “The local component of the service is implemented using Java **applications** that run on the **edge server** (and optionally, in the management system [“a key tool for the **NO [network operator]**”] and at the site of the **SP [service provider]**.”); see also my analysis of claim elements [1.1]-[1.3] below in §IX.B.1 above.

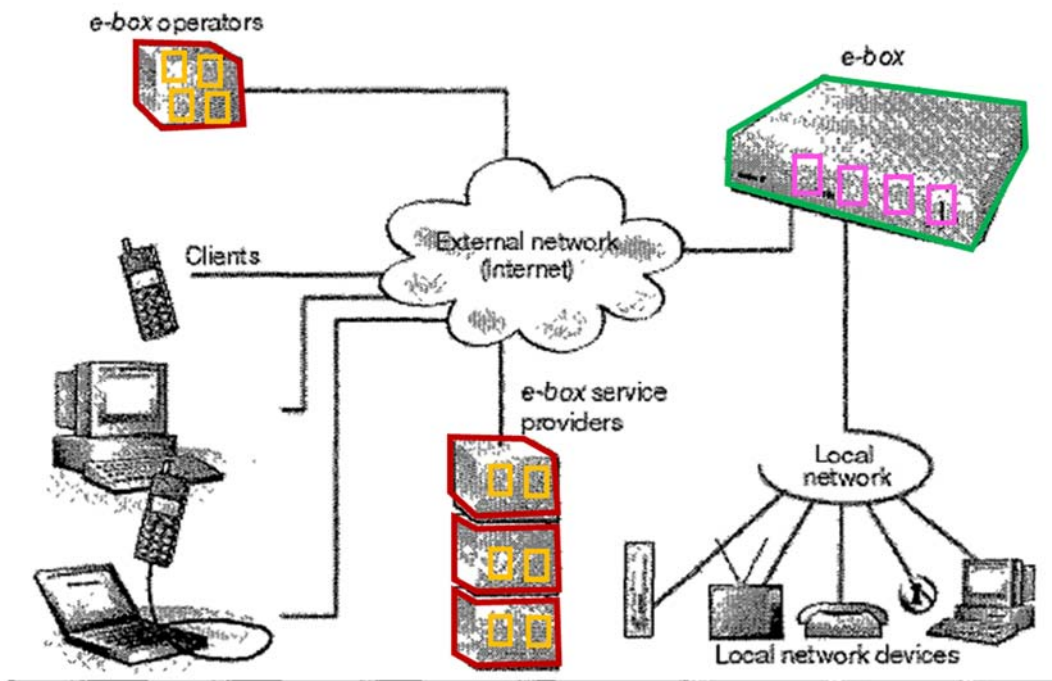


Figure 2  
The Ericsson e-service infrastructure.

EX1016 (Vasell 2<sup>nd</sup> Provisional), 5 (Figure 2 (which I have reproduced and annotated similar to Annotated FIG. 2 (above))).

401. Vasell discloses that the distributed “system software”, of each of the **local** and **remote** servers of the **service gateway system**, includes “a Java runtime in the form of a Java virtual machine (JVM)” and “an operating system” hosting the JVM which could be “[a]ny UNIX-like standard” OS, provided it has “fairly strong-resource control and protection mechanisms to ensure application integrity” such that “errors in software developed by different service providers” cannot result in the “applications... interfer[ing] with one another.”

402. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (description of the Java “[d]evelopment environment for **application software**” and

that “[s]ervices should be implemented as distributed **applications** ...”), 5 (Figure 2 (reproduced and annotated in my analysis of claim element [1.0] above in §IX.B.1 above), 7 (“The system software consists of: • an operating system; ... • server components—for example, Web servers ... • a Java runtime in the form of a Java virtual machine (JVM). Any UNIX-like standard system, including Open Source, can be chosen as an operating system. The system must have fairly strong resource-control and protection mechanisms to ensure application integrity.”), 8 (description of “**service applications**” and “development environment”), 9 (Figure 5, description of “*Edge server (ES)*”, “*Management system (MS)*”, “*network operator (NO)*”, “*Service provider (SP)*”); EX1004 (Vasell Patent), 18:23-30 (disclosing “**local**” and “**remote**” devices using a “Java Virtual Machine (JVM), or the like” or “multiple JVMs”).

403. Accordingly, my annotations in Annotated FIG. 2 (in ¶398 above) show, as disclosed in embodiments in Vasell, (i) a plurality of **service applications** (each including a plurality of distributed components, each a “boxlet within [a] cell”). Specifically, each component of a **service application** is respectively hosted on a Java Virtual Machine (JVM) running on the respective OS of the **service platform server(s) 22**, **network operator server(s) 24** (different connection options also illustrated (EX1004 (Vasell Patent), 6:65-7:3, EX1016 (Vasell 2<sup>nd</sup> Provisional), 5 (Figure 2), 9 (Figure 5)), and the **service provider equipment (servers) 34**. See

¶¶397-401 above and my citations to EX1004 (Vasell Patent) and EX1016 (Vasell 2<sup>nd</sup> Provisional) cited in these paragraphs; *see also* my analysis of claim elements [1.1]-[1.4] below in §IX.B.1 above.

404. **In sum**, Vasell discloses *a system* (a “**service gateway system**” that utilizes “distributed processing” across “servers or processors” connected to each other over “an Internet Protocol (IP) network such as the Internet”).

*[1.1] a programmable network device adapted to host a plurality of network device applications;*

405. Vasell discloses this limitation.

406. For example, Vasell discloses “**service platform server 22**” [*programmable network device*] is “an edge server” and “the hardware portion of a **service gateway** or **services platform**” (FIG. 2, [1.0] above in §IX.B.1 above), and states *it* “may provide service support for” “household users and residences..., an entire office building, an industrial plant, or even a campus-type organizational facility.” EX1004 (Vasell Patent), 5:61-6:8, 10:32-41. Vasell also states that the “**service platform server 22**” is “logically considered as a node associated with the network operator rather than the local area network [10] since... control for [its] maintenance, operation and support... is preferably performed by the network operator.” EX1004 (Vasell Patent), 6:9-20, 6:45-49.

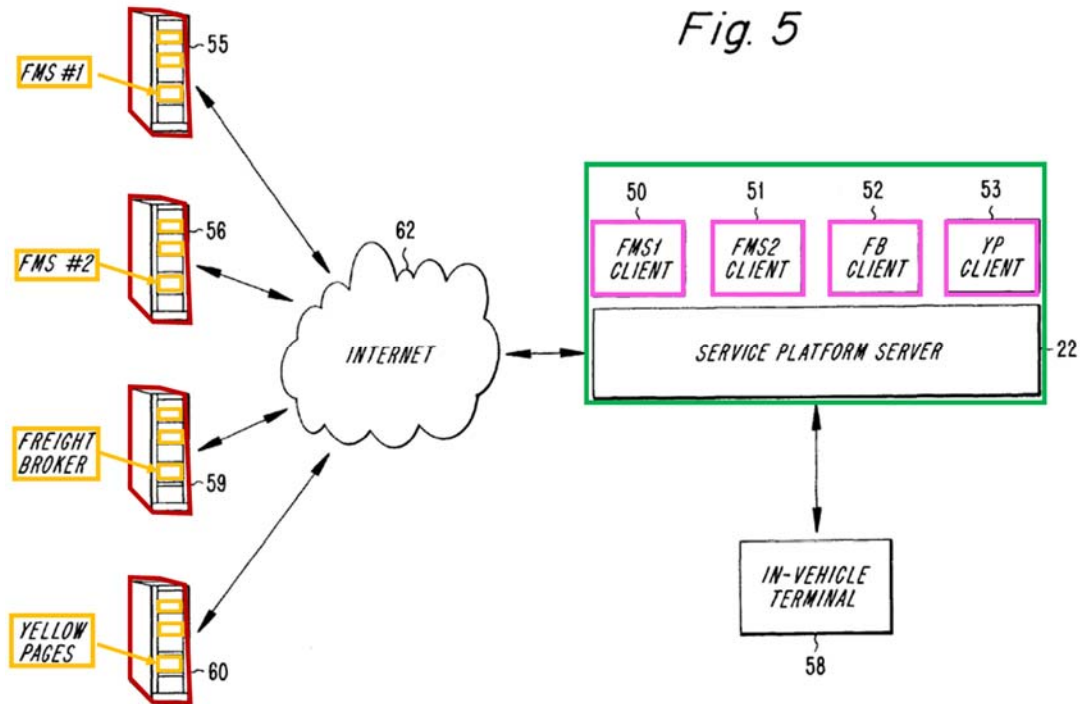
407. Vasell discloses “[t]he **service platform server 22** may contain **all or part of a software** application or **applications** [annotated **pink**] which are used to

implement one or more connectivity based services which have been requested and/or subscribed to by the end user” [*adapted to host a plurality of network device applications*]. EX1004 (Vasell Patent), 6:21-25; Annotated FIG. 2 (as I have annotated in claim element [1.0] above in §IX.B.1 above).

408. Vasell discloses many examples of the “distributed” **software service applications** “distributed” in the **service gateway system** (see my analysis of claim element [1.3] below in §IX.B.1 above), and of “**portions**”/**software components** thereof hosted on **service platform server 22**. *Id.*; EX1004 (Vasell Patent), 8:53-9:18 (in an example of “a service [being] implemented using software operating on two or more of the servers or processors associated with the [**service gateway system**] architecture illustrated in FIG. 2” [*i.e.*, “distributed processing techniques”]... a **third portion** of the **software** used to implement this service may reside on the **service platform server 22**”), 9:42-61 (disclosing an example of **service platform server 22** “multitask[ing] **software service applications** which implement the environmental control service **and** the telephony services”) (bold emphasis added).

409. For example, Vasell discloses, and illustrates in FIG. 5 (which I have reproduced and annotated below and refer to throughout my Declaration as “Annotated FIG. 5”), “**client software packages 50-53** which are running at any given time on the **service platform server 22**” [*programmable network device adapted to host a plurality of network device applications*]:

Fig. 5



EX1004 (Vasell Patent), 11:11-39, Annotated FIG. 5.

410. Vasell Patent further explains this process. EX1004 (Vasell Patent), 24:26:31 (claim 16: “downloading a portion of said service application to the subscriber’s service gateway unit ...”), 12:27-29 (“the software elements of the service gateway system may be distributed among various units of geographically separated equipment, as shown in FIG. 2”), 13:28-36 (e.g., “boxlet” within a “cell” of a “service application”), FIG. 6 (as I have annotated in my analysis of claim 14 in §IX.B.5 below including that “service applications 70-72 include the boxlets 64-69 within the cells 90-92” (13:28-36)).

411. Vasell 2<sup>nd</sup> Provisional provides more details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Edge servers ... run local applications ... Services should be

implemented as distributed **applications**. By executing over several infrastructure nodes, more complex services can be developed without the need for extremely powerful and complex **edge servers**. Distributed **applications** will considerably increase the technical life and reduce the cost of the **edge servers**.”), 7-8 (“A **service application** typically fulfills the role of a gateway between **servers** in the external network (from which the service is accessed and controlled) and **devices** in the local network.”), 8 (description of “**service applications**” and “system services”), 9 (Figure 5, “Box D, Distributed Services System Architecture” including, e.g., “The local component of the service is implemented using Java **applications** that run on the **edge server** (and optionally, in the management system [“a key tool for the **NO [network operator]**”] and at the site of the **SP [service provider]**.”)

412. **In sum**, Vasell discloses a programmable network device (“**service platform server 22**”) adapted to host a plurality of network device applications (adapted to host respective “**portions**”/software components of a plurality of distributed “**service applications**”).

***[1.2] a programmable cloud device adapted to host a plurality of cloud applications,***

413. Vasell alone, in view of Alves, or in view of Alves and Rellermeyer, discloses this limitation.

414. Vasell discloses “**service provider equipment 34**, which may include computer systems or servers, may be the initial repository for **software** used to

implement connectivity based services” [*programmable cloud device*], and identifies such “services are implemented upon request.” EX1004 (Vasell Patent), 7:4-20, 12:31-33; Annotated FIG. 2, *see* [1.0] above in §IX.B.1 above (“network operator server 24... connected to at least one **service provider equipment 34** via... Internet 26”); EX1004 (Vasell Patent), 11:40-46 (“[a]lthough frequently it will be desirable to interpose an intermediary network operator, there may be certain embodiments of the present invention where the network operator and the associated equipment may be eliminated” and **service provider servers 34** would instead provide maintenance and support functions (*see* ¶415 below).

415. Vasell also discloses “**network operator server 24 equipment** may include one or more servers, i.e., relatively powerful computers operating in conjunction with one or more secondary storage devices” [another *programmable cloud device*], which “handle[d] control for the operation, maintenance and support of the service[s] implementation” and “of the service platform servers 22 of various end users.” Vasell further discloses that “network operators using the **network operator servers 24** may act as intermediaries between service providers and the end users of those services.” EX1004 (Vasell Patent), 6:45-7:3, 6:16-20, 9:25-40, 11:40-46; *see* [1.0] above in §IX.B.1 above (“**service platform server 22**... connected to [a] **network operator server 24** via the Internet”), Annotated FIG. 2.

416. Vasell identifies “[v]arious corporate entities may function as network

operators according to the present invention, including, for example, communication utilities such as public telephone companies, communication companies, etc.” EX1004 (Vasell Patent), 6:54-58.

417. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 5-6 (Figure 2, “the *e-box* operator is the owner of the *e-box* network and, as such, is responsible for **managing** the individual units. From a technical viewpoint, this network is an operation and maintenance system.”), 7 (description of “system software”), 8 (description of “system services” and “**service application management**”), 9 (Figure 5, “*Network operator (NO)*: The entity that operates and maintains the system service registry and network of service access points and edge servers. The NO sells network access to service providers and can have roaming agreements with other NOs to provide a global service network. Examples of potential system service providers include network operators and Internet service providers.”; “*Management system (MS)*: The management system is a key tool for the NO. It allows the NO to control edge servers and the **service applications** that run on them.”; “*Service Provider (SP)*: The entity that furnishes services to end-users. The local component of the service is implemented using Java **applications** that run on the **edge server** (and optionally, in the management system [“a key tool for the NO [**network operator**]”] and at the site of the SP [**service provider**].”).

418. As I explained in my overview of Vasell (§VIII.A.1 above, ¶311 above

and in my Technology Background section (§VI.F.2 above (Distributed Systems – Clouds, ¶¶214-216 above)), based on Vasell’s disclosure that I discuss above for this claim element [1.2] above and claim element [1.0] above in §IX.B.1 above, in my opinion, POSITAs understood Vasell’s disclosed network of **remote servers** accessed “transparent[ly]” through the Internet, each respectively hosting respective **service application “portions”/software components** of distributed **service applications** on a respective JVM running on the respective OS on the **server’s** hardware to implement services through the **service gateway system**, to constitute a “*cloud*”, and each such server a *cloud device*. See EX1004 (Vasell Patent), 2:60-66 (“the ... implementation ... of services ... is transparent to both” “different service providers and the end user” and “[e]ach service may comprise a set of functionalities and logic which are implemented as software **service applications**.”); my analysis of claim elements [1.0] above, [1.2] above in §IX.B.1 above; §VIII.A.1 above (¶311 above); §VI.F.2 above (Distributed Systems – Clouds, ¶¶214-216 above *citing, e.g.*, EX1033 (Downing), 94; EX1052 (Sosinsky), xxv, 3, 45; EX1030 (Kashyap), ¶[0038]; EX1057 (Microsoft Computer Dictionary), 153).

419. As I discuss further in my analysis of claim element [1.3] in §IX.B.1 above, POSITAs understood Vasell discloses the **service provider equipment 34** of each service provider, and **network operator server 24** of each network operator, may each host, for example, respective “**portions”/software components** of distributed

service applications to implement each of a respective plurality of services [programmable cloud device adapted to host a plurality of cloud applications].

420. Vasell illustrates this with examples. See EX1004 (Vasell Patent), 8:53-9:24 (in an example of “a service [being] implemented using software operating on two or more of the servers or processors associated with the [service gateway system] architecture illustrated in FIG. 2 [*i.e.*, “distributed processing techniques”]..., a first portion of [the] software ... may reside permanently on the #1 service provider equipment 34” and “[a] second portion of the software ... may reside as part of the network operator server 24”), Annotated FIG. 2 (*see* claim element [1.0] above in §IX.B.1 above), 11:12-46 (in trucking industry example, associated freight company server (55, 56) software components, and/or associated “freight broker” 59, “yellow pages” 60, server software components), Annotated FIG. 5 (*see* claim element [1.1] above in §IX.B.1 above), 13:28-36 (*e.g.*, “service applications 70-72 include the boxlets 64-69 within the cells 90-92”), FIG. 6, 11:50-58 (describing an example of service provider equipment providing a plurality of services: *e.g.*, a “utility compan[y]” server (*see* ¶413 above) “implementing connectivity based services including automated meter reading, real time pricing ..., automated load control ...”), 2:64-66 (“Each service may comprise a set of functionalities and logic which are implemented as software service applications.”), 12:27-29 (“the software elements of the service gateway system may be distributed

among various units of geographically separated equipment, as shown in FIG. 2”).

421. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Services should be implemented as **distributed applications**. By executing over several infrastructure nodes, more complex services can be developed without the need for extremely powerful and complex **edge servers**. **Distributed applications** will considerably increase the technical life and reduce the cost of the **edge servers**.”), 5 (Figure 2), 7 (“A **service application** typically fulfills the role of a gateway between **servers** in the external network (from which the service is accessed and controlled) and **devices** in the local network.”), 9 (Figure 5, “Box D, **Distributed Services System Architecture**” including, *e.g.*, “The local component of the service is implemented using Java **applications** that run on the **edge server** (and optionally, in the management system [“a key tool for the **NO** [**network operator**]”] and at the site of the **SP** [**service provider**].”); *see also* my analysis of claim element [1.3] below in §IX.B.1 above).

422. **In sum**, in my opinion, Vasell discloses to a POSITA a programmable cloud device (“**service provider equipment 34**”, “**network operator server 24**”) adapted to host a plurality of cloud applications (adapted to host respective “**portions**”/**software components** of a plurality of distributed “**service applications**”).

423. To the extent the Patent Owner asserts that Vasell does not explicitly disclose its **service provider servers**, and **network operator server(s)**, as “*cloud*”

devices, as I discussed in the Motivation to Combine §IX.A above, it would have been obvious to POSITAs to implement these **servers** as a “cloud” as a mere design choice with a reasonable expectation of achieving well-known benefits. *See* §IX.A above (Motivation to Combine); *see also* my Technology Background §VI.F.2 above (Distributed Systems – Clouds) (¶¶214-216 above).

424. For example, Vasell emphasizes that, “[b]ecause the **service gateway system** is not limited to a particular technology or protocol, it can evolve freely with respect to access technology.” EX1004 (Vasell Patent), 7:12-14.

425. Moreover, as I discussed in the Motivation to Combine §IX.A above, and as established by Alves (EX1008), it was known to POSITAs that Vasell’s disclosed features (which it shared with OSGi)—*e.g.*, software “modularity” (*see* claim elements [1.0]-[1.2] above in §IX.B.1 above), and [1.3] below in §IX.B.1 above)); distributed execution of distributed **service applications** in a two-tier, multiple machine (**local network device** and **remote server device**), virtualized (JVM) architecture (*see* claim elements [1.0]-[1.2] above in §IX.B.1 above, and [1.3] below in §IX.B.1 above); dynamic software component updates without affecting the concurrent execution of other software components on the same JVMs on the same devices (*see* §IX.C.6 below (claim 18)); and dynamic resource (load) monitoring and control (*see* §IX.B.5 below (claim 14), §IX.C.3 below (claim 5), §IX.C.4 below (claim 7))—made Vasell’s **service gateway system** an “ideal” system

to implement in a cloud environment. *See* §IX.A above (Motivation to Combine); §VIII.A.2 above (Overview of the Prior Art – Alves) (*citing* EX1008 (Alves), xiii-xiv, 250, 261-263, 266-268, FIGS. 10.10-10.13); *see also* my Technology Background §VI.F.1 above (Distributed Systems – OSGi) (¶¶177-191 above, 213 above *citing, e.g.*, EX1008 (Alves), 250, 269, xiii-xiv).

426. Accordingly, in my opinion, POSITAs would have been motivated to select the known design choice of implementing Vasell’s **service provider servers**, and **network operator server(s)**, in, for example, a private/community/hybrid cloud as taught by Alves, where the **service gateway system** would continue to be **managed** by the network operator as disclosed by Vasell, with a reasonable expectation of achieving known and predictable benefits such as rendering “[t]he **system**... more scalable, because... we have a flexible setup, where we understand which **components** can be replicated and how best to partition the requests.” *Id.*; EX1008 (Alves), 263, 266, FIGS. 10.10, 10.12.

427. **In sum**, to the extent the Patent Owner asserts that Vasell does not explicitly disclose its **service provider servers**, and **network operator server(s)**, as “*cloud*” devices, in my opinion, Vasell in view of Alves discloses a programmable cloud device (“**service provider equipment 34**”, “**network operator server 24**”) adapted to host a plurality of cloud applications (adapted to host respective “**portions**”/**software components** of a plurality of distributed “**service applications**”).

428. Furthermore, to the extent that the Patent Owner asserts that, (i) in order to be a programmable “cloud” device, Vasell’s **remote management** functionality must include monitoring and control functionality for load (resource usage) expansion and contraction across Vasell’s **service provider servers** and **network operator server(s)**, and (ii) Vasell in view of Alves does not disclose *how* this is explicitly accomplished, as I discussed in the Motivation to Combine §IX.A above, these implementation details are disclosed in Rellermeyer (EX1011) as being accomplished “using features already provided by OSGi” specifications (which Vasell’s disclosure was incorporated into). *See* §IX.A above (Motivation to Combine); §VIII.A.2 above (Overview of the Prior Art – Alves) (*citing* EX1008 (Alves), 266-268, FIGS. 10.11, 10.13); §VIII.A.4 above (Overview of the Prior Art – Rellermeyer) (*citing* EX1011 (Rellermeyer), 2-4).

429. Moreover, I discussed in the Motivation to Combine §IX.A above, POSITAs would have been motivated to apply Rellermeyer’s teachings of the OSGi **management software** monitoring and controlling load expansion/contraction across multiple **cloud servers** as a known improvement to OSGi **distributed systems** with a high likelihood of achieving known benefits including, for example, turning Vasell’s connectivity-based services “into elastic units of deployment which can be migrated and replicated across multiple **cloud nodes** to increase the availability and reliability of the administered service” given POSITAs’ knowledge that OSGi’s and Vasell’s

shared features made OSGi (and Vasell) an “ideal” system to implement in a well-known “cloud” environment (see ¶425 above). *Id.*; see also §VIII.A.4 above (Overview of the Prior Art – Rellermeyer) (citing EX1011 (Rellermeyer), 2-5); EX1016 (Vasell 2<sup>nd</sup> Provisional), 9 (“The [Network Operator] sells network access to service providers and can have roaming agreements with other NOs to provide a global service network.”).

430. **In sum**, to the extent the Patent Owner asserts that (i) in order to be a programmable “cloud” device, Vasell’s **remote management** functionality must include monitoring and control functionality for load (resource usage) expansion and contraction across Vasell’s **service provider servers** and **network operator server(s)**, and (ii) Vasell in view of Alves does not disclose *how* this is explicitly accomplished, in my opinion, Vasell in view of Alves and Rellermeyer discloses a programmable cloud device (“**service provider equipment 34**”, “**network operator server 24**”) hosting a plurality of second network applications (hosting respective “**portions**”/**software components** of a plurality of distributed “**service applications**”).

***[1.3] wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form a distributed application;***

431. Vasell alone, or in view of Alves, discloses this limitation.

432. Vasell discloses distributed applications (distributed **service**

applications), implementing connectivity-based services in the service gateway system.

433. As an example of distributed service applications in the service gateway system, Vasell discloses, with respect to Annotated FIG. 2 (see my analysis of claim element [1.0] above in §IX.B.1 above), that:

For example, a first portion of software associated with the environmental service described above may reside permanently on the #1 service provider equipment 34. A second portion of the software used to implement environment control in the end user's residence may reside as part of the network operator server 24. Likewise, a third portion of the software used to implement this service may reside on the service platform server 22. The distribution of software used to implement any particular service will, of course, depend upon the nature of the service, the nature of the software, and the bandwidth associated with various information transfer needed to implement the desired service... Rather than being distributed over all three pieces of equipment, those skilled in the art will appreciate that software may be distributed between the network operator server 24 and the service platform server 22, or between the service provider equipment 34 and the service platform server 22, or any other combination of processors deemed desirable.

EX1004 (Vasell Patent), 8:53-9:24.

434. The Vasell Patent further explains the implementation of services for end users with distributed service applications, each having a component

(“portion”) installed on a **service platform server**, and a **component** (“portion”) installed on a **network operator server** and/or **service provider equipment**. EX1004 (Vasell Patent), 2:64-3:1 (“Each service may comprise a set of functionalities and logic which are implemented as **software service applications**. The **service application** may be **distributed** among various pieces of equipment which are geographically separated.”), 24:49-51 (“wherein said **service application** is a **distributed** software, that is **distributed** across **processors** in addition to said **service gateway unit**”), 12:27-29 (“the **software elements** of the **service gateway system** may be **distributed** among various units of geographically separated equipment, as shown in FIG. 2”), 13:28-36 (*e.g.*, “**service applications** 70-72 include the boxlets 64-69 within the cells 90-92”), FIG. 6.

435. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Services should be implemented as **distributed applications**. By executing over several **infrastructure nodes**, more complex services can be developed without the need for extremely powerful and complex **edge servers**. Distributed **applications** will considerably increase the technical life and reduce the cost of the **edge servers**.”), 5 (Figure 2), 7 (“A **service application** typically fulfills the role of a gateway between **servers** in the external network (from which the service is accessed and controlled) and **devices** in the local network.”), 8 (description of “**service applications**” and “system services”), 9 (Figure 5, “Box D, Distributed Services

System Architecture” including, *e.g.*, “The local component of the service is implemented using Java **applications** that run on the **edge server** (**and** optionally, in the management system [“a key tool for the **NO** [**network operator**]”] and at the site of the **SP** [**service provider**].”).

436. Another example of **distributed service applications** in the **service gateway system**, where each distributed **application** has a component at a **client** location and a component at a **service provider** location, is shown and described in Vasell’s Annotated FIG. 5 (*see* my analysis of claim element [1.1] above in §IX.B.1 above). Vasell discloses, using a trucking industry example, **mobile service platform server 22** (“outfitted” on a truck) has various “service **software client packages**” (**50-53**) loaded thereon (*e.g.*, in a JVM running on the **server** OS) that communicate with **associated software components** loaded on **service provider servers** (**55, 56, 59, 60**) (*e.g.*, in respective JVMs running on each **server** OS) to provide services to trucking industry users. EX1004 (Vasell Patent), 11:13-46, Annotated FIG. 5, 2:64-66.

437. In this example, Vasell discloses “two different freight companies, FMS #1 and FMS #2,” may each use **mobile service platform server 22** for services by having respective **software** on respective **service provider servers** (**55, 56, 59, 60**) communicate with **associated software packages** (**50-53**) running on the **mobile server** to “coordinate operations of the truck” and for inter-application information exchanges (*e.g.*, “**client software package**” may provide the **freight company server**

(55, 56) software component of a distributed application with “information regarding the current location of the truck and its anticipated arrival time of delivery”; “freight broker” 59, and/or “yellow pages” 60, server software components may provide information required by “client software packages”; etc.).

*Id.*

438. Vasell 2<sup>nd</sup> Provisional provides details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Services should be implemented as distributed applications. By executing over several infrastructure nodes, more complex services can be developed without the need for extremely powerful and complex edge servers. Distributed applications will considerably increase the technical life and reduce the cost of the edge servers.”), 7 (“A service application typically fulfills the role of a gateway between servers in the external network (from which the service is accessed and controlled) and devices in the local network.”), 9 (Figure 5, “Box D, Distributed Services System Architecture” including, *e.g.*, “The local component of the service is implemented using Java applications that run on the edge server (and optionally, in the management system [“a key tool for the NO [network operator]”] and at the site of the SP [service provider].”).

439. **In sum**, Vasell discloses a distributed application (a distributed service application implementing a connectivity-based service in the service gateway system) that is composed of at least one of the plurality of first network applications

in the **programmable network device** and at least one of the plurality of **second network applications** in the **programmable cloud device** (that is composed of at least one of the **software components** (each a “boxlet” within a “cell”) in the **service platform server** (*see also* claim element [1.1] above in §IX.B.1 above) and at least one of the **software components** (each a “boxlet” within a “cell”) in a **remote (cloud) server** (*see also* claim element [1.2] above in §IX.B.1 above).

440. Vasell also discloses that the underlying **software** components (each a “boxlet” within a “cell”) that collectively form each of these examples of distributed **service applications** *are in secure communication with each other to form such distributed applications* to POSITAs. I discuss three reasons next.

441. First, as I discussed in my analysis of claim element [1.0] above in §IX.B.1 above, Vasell discloses that the respective **software** components of each of these distributed **service applications** (*see, e.g.*, ¶¶431-438 above) would be respectively hosted on a respective “Java Virtual Machine (JVM)” running on the respective OSs of “**local**” and “**remote**” servers (the **service platform server**, the **network operator server(s)**, and the **service provider equipment**) in the **service gateway system**.

442. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“[s]ervices should be implemented as distributed **applications** ...”, “*Development environment for application software*”: The development environment must follow

the ‘write-once, run-everywhere’ maxim and should be based on **Java standards**. New **applications** will interact with the **e-service infrastructure** through **Java application program interfaces (API)** that comply with mainstream **Java** development. By leveraging the **Java** development, the **application software** environment can be taken to a **higher level of abstraction**, allowing non-specialists to develop **service applications** more easily.”), 7 (“The system software consists of: • an operating system; ... • server components-for example, Web servers ... • a **Java runtime in the form of a Java virtual machine (JVM)**. Any UNIX-like standard system, including Open Source, can be chosen as an operating system. The system must have fairly strong resource-control and protection mechanisms to ensure application integrity.”), 8 (description of “**service applications**” including “The service platform is a **Java environment**; that is, a service is basically a **Java application**, albeit a somewhat restricted one. The type of **Java application** allowed by the service platform is called a *boxlet*, which consequently, is a piece of **Java** code that implements a service.”, and “*development environment*” including “Boxlets are created using a standard **Java** development environment.”), 9 (“The local component of the service is implemented using **Java applications** that run on the **edge server** (and optionally, in the management system [“a key tool for the **NO** [network operator]”] and at the site of the **SP** [service provider].”); EX1004 (Vasell Patent), 18:23-30 (disclosing “**local**” and “**remote**” servers using a “Java Virtual

Machine (JVM), or the like” or “multiple JVMs”).

443. Second, Vasell emphasizes that “Applicants consider **secure** implementation of these connectivity based services to be important” and, “[s]ince various service providers, network providers and, possibly, end users, will share infrastructure, it is important that information and **communication associated with each implemented connectivity based service**”—*i.e.*, communications between the respective software components that collectively form each of these distributed **service applications** to implement each such service—“**be secure.**” EX1004 (Vasell Patent), 11:59-64.

444. Specifically, Vasell expressly states that “Applicants envision **widespread usage of... security techniques including data encryption**, various authorization and identity verification techniques, etc. to be provided **so that the implementation, monitoring ... associated with these services is robustly defended against intrusion and manipulation.**” *Id. see also* EX1016 (Vasell 2<sup>nd</sup> Provisional), 3 (“Consumer Requirements of E-Services” include “*Security*: Consumers **do not tolerate** invasion of privacy or hacking into mission-critical services by strangers or neighbors.”), 6 (“*Security*: The nature of many of the services that can be expected to be based on the *e-box* system, and the fact that the system will be shared by multiple service providers, **make security a major issue.**”).

445. Third, consistent with the well-known sandboxing of the run-time environment of JVMs running on the host OSs of **service gateway system servers**, Vasell discloses that each of the underlying software components that collectively form each of the distributed **service applications** is “designated into various categories having different degrees of access restriction”, where “access rules” (**administrator-defined** for each application’s cell (which “represent[s] the resources available to” the particular application) and for application cell “groups”) are **controlled** and **enforced** using, in part, “inter-cell communication... controlled interfaces” called “gates” (**managed** by “**gate manager 96**”), including to “limit[], monitor[], and control[]” “access of the cells.” *Id.*; EX1004 (Vasell Patent), 11:64-12:10, 4:58-65, FIG. 6, 13:28-36 (e.g., “**service applications 70-72** include the boxlets 64-69 within the cells 90-92”), 14:2-7; EX1016 (Vasell 2<sup>nd</sup> Provisional), 8 (description of “**service applications**”, “*communication between applications*” and “*system services*”), Figure 4; *see also* my Technology Background §VI.D.3 above – Java Virtual Machines, Java Applications, and Digital Signature Implementations, ¶¶130-134 above.

446. Accordingly, given at least these three (3) exemplary disclosures in Vasell (*see* ¶¶439-445 above), although Vasell uses different words than the ’823 Patent, in my opinion, a POSITA would have understood Vasell as disclosing its distributed **service applications** being formed by the respective underlying Java

software components (each a Java “boxlet” within a “cell”) securely and “transparent[ly]” communicating with each other using well-known “data encryption” protocols, **managed** “gates” enforcing cell-specific “access rules”, and the respective virtualized, sandboxed, and “higher level of abstraction” environment of a JVM running on the respective OS of the **service platform server** or **remote (cloud) server** on which they are hosted [*in secure communication with each other to form distributed applications*] in the same way as the ’823 Patent. *Id.*

447. Compare the three (3) exemplary disclosures in Vasell (*see* ¶¶439-445 above) and EX1004 (Vasell Patent), 2:60-3:1 (“the ... implementation ... of services ... is transparent to both” “different service providers and the end user” and “[e]ach service may comprise a set of functionalities and logic which are implemented as software **service applications**.”) *with* §VII.A above (Summary of the ’823 Patent) (*citing, e.g.,* EX1001 (’823 Patent), 10:44-50 (“[i]t is important... that the **fxDeviceApp** 302b and **fxCloudApp** 304a could use any protocol to communicate with each other”), 20:49-62 (“Virtual Fabric (fxVF)... may use [developers’] own communication protocols between **fxDeviceApps**... and **the associated fxCloudApp**” and “provide[s] a secure routing mechanism”), FIG. 15, 20:64-21:18 (“fxVF follows the security policies determined by the network administrator for inter-application communications”), 21:36-44 (same), 13:39-42 (“enables transparent switching of **application** and system messages of a dApp between the

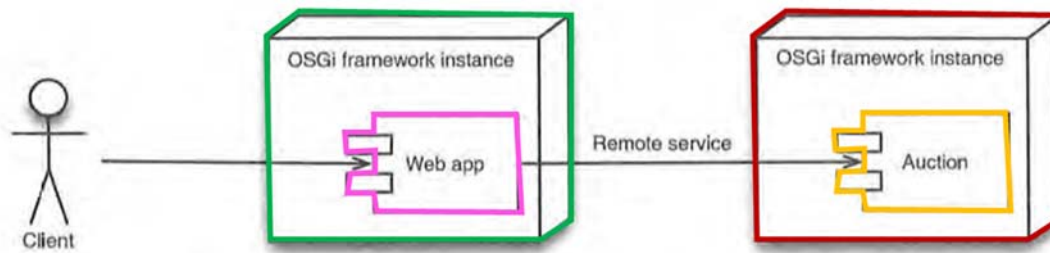
fxCloud 304 and the fxDevice 302”), 11:62-67 (“The fxCloud 304 and the fxOS 302a create a virtual fabric (fxVF) for messaging between applications ... The messaging complexity may be abstract for the developer.”), 7:58-62 (“The Distributed Software Defined Network (dSDN) disclosed herein is an end-to-end architecture that enables secure and flexible programmability across a network with full lifecycle management of services and applications (fxApp).”), 2:32-36 (same)).

448. **In sum**, in my opinion, a POSITA would have also understood that Vasell discloses forming distributed **service applications** by their respective underlying Java software components being in secure communication with each other (the respective underlying Java software components, each a Java “boxlet” running in the respective virtualized, sandboxed, and “higher level of abstraction” environment of a JVM running on the respective OS of the **service platform server** or **remote (cloud) server** on which they are hosted, securely and “transparent[ly]” communicate with each other using well-known “data encryption” protocols, **managed** “gates” enforcing cell-specific “access rules”, and Java APIs).

449. To the extent that the Patent Owner asserts that Vasell does not explicitly disclose **secure** communications between the underlying Java software components of Vasell’s distributed **service applications**, in my opinion, it would have been obvious to POSITAs to **securely** implement these inter-application communications, between **distributed** Java software components respectively

running in a virtualized (JVM) environment (respectively running on a host OS of a **service gateway system server**), as a mere design choice with a reasonable expectation of achieving the benefits expressly disclosed by Vasell (*see, e.g.*, ¶444 above) and other well-known benefits.

450. As I discussed in the Motivation to Combine §IX.A above, Alves (EX1008) discloses that these secure communication technical implementation details were later standardized by OSGi in the Remote Service Specification (EX1028), including, for example, that communications between a **locally installed** “**bundle** that’s exporting a remote service in one OSGi framework instance and [a **remotely installed**] **bundle** that’s importing the service in another [OSGi] framework instance” would, for example, use well-known protocols “such as RMI (Remote Method Invocation), SOAP (Simple Object Access Protocol), and REST (Representational State Transfer)”, each of which a POSITA would understand, as explained in the Technology Overview – Distributed Systems – OSGi section (§VI.F.1 above), as providing secure communications between **distributed** software components “so that the implementation, [and] monitoring... associated with” **distributed-service application**-provided-services “is robustly defended against intrusion and manipulation” (EX1004 (Vasell Patent), 11:59-64) and to satisfy “consumer requirements” of such services (EX1016 (Vasell 2<sup>nd</sup> Provisional), 3 (“*security*”):



**Figure 10.9** An auction system partitioned into two separate OSGi framework instances, communicating through an OSGi remote service

EX1008 at Figure 10.9 (which I reproduce and annotate above).

451. Figure 10.9 shows an example of an auction system OSGi-distributed application in which a “web app” bundle installed in an OSGi environment (including a JVM) hosted on the OS of a local machine securely communicates with an “auction” bundle installed in an OSGi environment (including a JVM) hosted on the OS of a business tier machine); Motivation to Combine §IX.A above (citing §VIII.A.2 above (Overview of Alves, citing EX1008 (Alves), xiv, 249-250, 252-255, 261-263, Figure 10.9 Figure 10.10; and citing §VI.F.1 above (Technology Background – OSGi section) (¶¶205-212 above) (citing id.; EX1046 (Oasis - SOAP Security Specification), 1-2, 7-8, 13-15; EX1047 (OWASP - WSS Security Specification), 1-2, 4; EX1048 (OWASP - REST Security Specification), 1-2; EX1053 (JSSE Guide), 3-5, 68; EX1054 (Java RMI Overview), 1; EX1055 (Java RMI Socket Overview), 1; EX1056 (Java RMI SSL Overview), 1)).

452. **In sum**, in my opinion, to the extent the Patent Owner asserts that Vasell does not explicitly disclose **secure** communications between the underlying

Java software components of Vasell’s **distributed service applications**, Vasell in view of Alves and the knowledge, skill, and creativity of a POSITA discloses forming distributed **service applications** by their respective underlying Java software components being in **secure** communication with each other (the respective underlying Java software components, each a Java “boxlet” running in the respective virtualized, sandboxed, and “higher level of abstraction” environment of a JVM running on the respective OS of the **service platform server** or **remote (cloud) server** on which they are hosted, **securely** and “transparent[ly]” communicate with each other using a well-known protocol (*e.g.*, RMI, SOAP or REST) that a POSITA would understand as providing secure communications between **distributed** Java software components and to achieve known increased security benefits (including those disclosed by Vasell)).

*[1.4] wherein the plurality of network device applications and plurality of cloud applications device form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware.*

453. Vasell discloses this limitation.

454. As an initial matter, a POSITA would have understood the importance of well-defined APIs and the benefits of standardizing them. *See* §§VI.A, VI.B, VI.D.3, and VI.F.1 above.

455. Vasell discloses that the hardware-independence of the application programming interfaces (APIs) to program the **service applications** is important due to the market-driven “open” nature of the **service gateway system**.

456. For example, Vasell discloses that “[t]he market seems to favor an **open** e-service platform that allows **several independent** service providers to **share the same infrastructure** in order to reduce cost. In a likely market scenario, a third-party network operator will assume responsibility for operating and maintaining a complete industry network (Figure 1).” EX1016 (Vasell 2<sup>nd</sup> Provisional), 3; *id.*, (“the strength of the network operator depends on the number of service providers using the network, and an operator will therefore try to keep the platform as **open** as possible”), *id.*, 9 (“[A] new **e-service infrastructure** is required—an **open** platform that can be used by **several independent** service providers ... Anticipating these ... demands, Ericsson has developed the *e-box* system.”). (original emphasis).

457. Vasell Patent (EX1004) adds that “[s]ervice gateways according to the present invention are preferably **open** platforms to allow for the development of **service applications** by **third party developers**” using “standard communication protocols and programming languages”, “service gateways according to the present invention are intended to be horizontal in nature so that a plurality of service providers can share the same platform infrastructure”, and “service gateway compatibility potentially opens **third-party markets** for... services.” EX1004

(Vasell Patent), 3:10-19, 5:30-33, 10:18-23, 2:60-66 (“[t]he **service gateway system** according to the present invention facilitates the development... of services... [by] **different services providers**” where “[e]ach service... [is] implemented as software **service applications**.”), 13:37-43.

458. Given the “open” nature of the **service gateway system**, Vasell discloses, for example, that “[t]he development environment [for **application software**] **must** follow the ‘**write-once, run-everywhere**’ **maxim** and should be based on Java standards. New **applications** will interact with the **e-service infrastructure** through **Java application program interfaces (API)** that **comply with mainstream Java development**. **By leveraging the Java development, the application software environment can be taken to a higher level of abstraction**, allowing nonspecialists to develop **service applications** more easily” [*unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware*]. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4; *see also id.*, 8 (“**Boxlets** are created using a **standard** Java development environment. For instance, the Java development kit (JDK) from Sun Microsystems can be used as well as other development environments. The only parts that are specific to **boxlet** development are the libraries that contain APIs for the **main services** and **system services** layers.”).

459. As discussed in §VII.B above (Prosecution History of the '823 Patent), I have been instructed by counsel to assume that the “wherein the plurality of network device applications and plurality of cloud applications **device** form ...” (in this claim element [1.4]) is a typographical error, and to also interpret this claim limitation as reciting “wherein the plurality of network device applications and plurality of cloud applications form...”.

460. Moreover, given the “open” nature of the **service gateway system**, Vasell discloses that “[t]he **service gateway system** according to the present invention facilitates **the development, implementation, operation and maintenance of services** in an integrated manner **so that the interface** between different **services providers** and the end user **is transparent** to both” where “[e]ach service” is “implemented as **software service applications** ... **distributed** among various pieces of equipment which are geographically separated.” EX1004 (Vasell Patent), 2:60-3:1; *id.*, 5:15-35 (identifying “desirable characteristics of the **service gateway [system]** design” as “the ability for a service gateway to be **managed remotely** in as many ways as possible” and “**compatib[ility] to established communications standards, protocols, interface specifications and technologies**” as it “opens **third-party markets** for **service gateway hardware and services.**” [another exemplary disclosure of *unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality*

*of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware].*

461. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (disclosing that the **service gateway system** “infrastructure **must be flexible, open and modular**, to accommodate **a range of communication standards and protocols.**”), *id.*, at 4, 6 (“[t]o the greatest possible extent, *the e-box* system is based on **standard technologies**, so as to allow **third-party markets for hardware and services** to evolve”, and where such “[s]ervices should be implemented as **distributed applications**” since, “[b]y executing over **several infrastructure nodes**, more complex services can be developed **without the need for extremely powerful and complex edge servers.**”); *id.*, at 8-9 (disclosing (i) “[*t*he *e-box* implementation does **not** stipulate the type of external network to be used in an *e-box* system, nor does it specify what access technology is to be used”; (ii) “[a]s far as **software** is concerned, adapting *the e-box* to a new form of local network is a matter of integrating the necessary drivers into a local network system service”; and “[o]n the hardware side, it may entail the development of a suitable network interface card.”).

462. Accordingly, in my opinion, a POSITA would have understood that Vasell discloses that the plurality of network device applications and plurality of cloud applications (the respective underlying Java software components in secure communication with each other to form distributed **service applications**, *see* claim

elements [1.1]-[1.3] above (§IX.B.1 above)) form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware. Specifically, Vasell discloses this limitation [1.4] above (§IX.B.1 above) as follows: (1) the distributed **service applications** are naturally modular; (2) the respective underlying Java software components, of such distributed **service applications**, use standardized communication protocols to communicate with each other; (3) standard Java APIs are used for **their** development; (4) “flexible”, “transparent”, and standards-compatible interfaces are used for **their lifecycle management** (see claim element [19.5] below (§IX.B.7 below)) in the **service gateway system**; and (5) these standard development and **management** APIs are usable regardless of the respective **service platform servers’** hardware and **remote (cloud) servers’** hardware on which the respective JVMs run on the respective OSs; see claim elements [1.1]-[1.3] above (§IX.B.1 above)).

463. **In sum**, in my opinion, Vasell discloses to a POSITA that the plurality of network device applications and plurality of cloud applications (the respective underlying Java software components in secure communication with each other to form distributed **service applications**, see claim elements [1.1]-[1.3] above (§IX.B.1 above)) form unified capabilities enabling a plurality of upper layer application

programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware (e.g., (1) the distributed **service applications** are naturally modular; (2) the respective underlying Java software components, of such distributed **service applications**, use standardized communication protocols to communicate with each other; (3) standard Java APIs are used for **their** development; (4) “flexible”, “transparent”, and standards-compatible interfaces are used for **their lifecycle management** (see claim element [19.5] below (§IX.B.7 below)) in the **service gateway system**; and (5) these standard development and **management** APIs are usable regardless of the respective **service platform servers’** hardware and **remote (cloud) servers’** hardware on which the respective JVMs run on the respective OSs; see claim elements [1.1]-[1.3] above (§IX.B.1 above)).

**2. Claim 2: The system of claim 1, wherein the programmable network device and the programmable cloud device each include an access network interface unit configured to send and receive communications and a processor with a memory associated with the network interface unit.**

464. Vasell discloses this limitation.

465. As I explained in my analysis of claim elements [1.1]-[1.2] above in §IX.B.1 above), Vasell discloses the “**service platform server 22**” [*programmable network device*] includes “access technology” to send and receive communications in the LAN (e.g., “Ethernet cable,... modem or other gateway connection”) and over

the “Internet or other communication network” (e.g., “wireless... connection), and discloses that the “**network operator server**” and “**service provider equipment**” [*programmable cloud device*] also include “access technology” to communicate over, and be “connected to”, “an Internet protocol (IP) network such as the Internet 26 or other network technology... [e.g.] wireless”) [*the programmable network device and the programmable cloud device each include an access network interface unit configured to send and receive communications*]. EX1004 (Vasell Patent), 5:64-6:8, 6:62-7-14.

466. Vasell also discloses that the “**cell manager**”, managed via the “**management system service**” of “system services layer 110”, which “is implemented in the form of the boxlets[] executing within cells”, monitors resource usage, including “CPU time, memory”, on each of these devices by the **service applications**, and identifies that the “distributed **processing** techniques... is implemented using software operating on two or more of the servers or **processors** associated with the architecture illustrated in FIG. 2” [*a processor with a memory associated with the access network interface unit*]. EX1004 (Vasell Patent), 20:45-49, 22:34-37, 20:23-29, FIG. 6, 14:15-17, 20:55-58, 22:49-53, 8:53-58, 9:13-19; Annotated FIG. 2 (*see claim element [1.0] above in §IX.B.1 above*).

467. Accordingly, in my opinion, a POSITA would have understood that Vasell discloses that each of the **service platform servers** and **remote (cloud) servers**

of the **service gateway system** also includes “*a processor with a memory associated with the access network interface unit.*” *Id.*

**3. Claim 12: The system of claim 1, further comprising a virtual fabric which provides a secure communication layer for said at least one of the plurality of network device applications and said at least one of the plurality of cloud applications.**

468. Vasell alone, or in view of Alves, discloses the limitations of claim 12.

469. As I explained in my analysis of claim element [1.3] above (§IX.B.1 above), Vasell alone, or in view of Alves, discloses forming distributed **service applications** by their respective underlying Java software components being in **secure** communication with each other (the respective underlying Java software components, each Java “boxlet” running in the respective virtualized, sandboxed, and “higher level of abstraction” environment of a JVM running on the respective OS of the **service platform server** or **remote (cloud) server** on which they are hosted, **securely** and “transparent[ly]” communicate with each other using a well-known protocol (*e.g.*, RMI, SOAP or REST) that a POSITA would understand as providing secure communications between **distributed** Java software components and to achieve known increased security benefits (including those disclosed by Vasell)). *See* my analysis of claim element [1.3] above (§IX.B.1 above).

470. For at least the same reasons that I discussed above in my analysis of claim element [1.3] above (§IX.B.1 above), although Vasell uses different words

than the '823 Patent, in my opinion, a POSITA would have understood Vasell, alone or in view of Alves, as disclosing, with the same, or an equivalent, structure as the '823 Patent, that the **service gateway system** includes *a virtual fabric which provides a secure communication layer* over which the respective underlying Java software components (each a Java “boxlet” within a “cell”), of Vasell’s distributed **service applications**, securely and “transparent[ly]” communicate with each other using well-known “data encryption” protocols, **managed** “gates” enforcing cell-specific “access rules”, and the respective virtualized, sandboxed, and “higher level of abstraction” environment of a JVM running on the respective OS of the **service platform server** or **remote (cloud) server** on which they are hosted. *See* my analysis of claim element [1.3] above (§IX.B.1 above) (*citing* EX1016 (Vasell 2<sup>nd</sup> Provisional), 3-4, 6-9; EX1004 (Vasell Patent), 18:23-30, 11:59-64, 11:64-12:10, 4:58-65, FIG. 6, 13:28-36; EX1008 (Alves), xiv, 249-250, 252-255, 261-263, Figure 10.9, Figure 10.10)

471. The Claim Construction briefs provide related details. See §VIII.B.1 above (Claim Construction section – “virtual fabric”) (Patent Owner identifying “an abstraction layer for applications to communicate with each other”, as described in EX1001 ('823 Patent), 13:33-39, 20:36-41, 11:57-62 (which is a shared specification with the '871 Patent and the '095 Patent), as the structure corresponding to the recited function and the portions of the specification describing this structure.

EX1070 (Patent Owner Claim Construction Brief), 9-10; EX1071 (Petitioner Claim Construction Brief), 12-13).

472. To the extent that the Patent Owner asserts or the Board finds that this is a means-plus function limitation, the “way” that Vasell discloses forming distributed **service applications** by their respective underlying Java software components being in secure communication with each other through a virtual fabric is that the respective underlying Java software components, each a Java “boxlet” running in the respective virtualized, sandboxed, and “higher level of abstraction” environment of a JVM running on the respective OS of the **service platform server** or **remote (cloud) server** on which they are hosted, securely and “transparent[ly]” communicate with each other using well-known “data encryption” protocols, managed “gates” enforcing cell-specific “access rules”, and Java APIs.

473. In my opinion, a POSITA would have understood that this “way” that Vasell discloses forming distributed **service applications** is substantially similar (at least by being one example thereof) of the “way” that “at least one of the plurality of first network applications in the programmable network device and at least one of the plurality of second network applications in the programmable cloud device are in secure communication with each other through a virtual fabric to form a distributed application” is described in EX1001 (’823 Patent), 13:33-39, 20:36-41, 11:57-62.

474. Also, in my opinion, a POSITA would have understood that the “result” of Vasell’s disclosure of forming distributed **service applications** by their respective underlying Java software components being in secure communication with each other through a virtual fabric is also substantially similar to the “result” in the ’823 Patent’s disclosure, or thus that the structure in Vasell for performing the function of “form[ing] a distributed application” is at least equivalent to the Patent Owner identified structure in the specification of the ’823 Patent.

475. **In sum**, in my opinion, a POSITA would have understood that Vasell, alone or in view of Alves, as disclosing, with the same, or an equivalent, structure as the ’823 Patent, that the **service gateway system** includes *a virtual fabric which provides a secure communication layer* over which the respective underlying Java software components (each a Java “boxlet” within a “cell”), of Vasell’s distributed **service applications**, securely and “transparent[ly]” communicate with each other using well-known “data encryption” protocols, **managed** “gates” enforcing cell-specific “access rules”, and the respective virtualized, sandboxed, and “higher level of abstraction” environment of a JVM running on the respective OS of the **service platform server** or **remote (cloud) server** on which they are hosted.

4. **Claim 13: The system of claim 1, further comprising: a second programmable network device which includes at least one of the plurality of network device applications which is the same as at least one of the plurality of network device applications in the programmable network device**

**and where each of the network device applications can communicate directly.**

476. Vasell alone, or in view of Alves, discloses the limitations of claim 13.

477. Vasell discloses that that “the **service gateway system** is accessed by and shared with multiple service providers and service gateway users”, and that the “service gateway network may potentially range from **several hundred thousand to millions of service gateway units.**” EX1004 (Vasell Patent), 4:51-54, 5:16-20. A POSITA would have understood this to disclose a (rather large) plurality of devices.

478. In my opinion, a POSITA reading Vasell understood that it discloses, including based on the examples of “service providers” and examples of “connectivity based services” being provided to “various end users” (residences and/or businesses), different **service platform servers** running the same distributed **application software** “**portion**” thereon so that the same connectivity-based service is provided to different end users via the service gateway system. *Id.*; EX1004 (Vasell Patent), 6:45-49, 3:52-57, 10:17-38, FIG. 3, 11:47-58.

479. As I explained in my analysis of claim element [1.3] above in §IX.B.1 above), Vasell alone, or in view of Alves, discloses forming distributed **service applications** by their respective underlying software components being in **secure** communication with each other (the respective underlying Java software components, each Java “boxlet” running in the respective virtualized, sandboxed,

and “higher level of abstraction” environment of a JVM running on the respective OS of the **service platform server** or **remote (cloud) server** on which they are hosted, **securely** and “transparent[ly]” communicate with each other using a well-known protocol (*e.g.*, RMI, SOAP or REST) that a POSITA would understand as providing secure communications between **distributed** Java software components and to achieve known increased security benefits (including those disclosed by Vasell). *See* my analysis of claim element [1.3] above in §IX.B.1 above).

480. For at least the same reasons that I discussed in my analysis of claim element [1.3] above (§IX.B.1 above), in my opinion, a POSITA would have understood Vasell, alone or in view of Alves, as disclosing that, in implementing **secure** communications between the respective underlying Java software components, which form the distributed **service applications** and which use well-known communication protocols (*e.g.*, RMI, SOAP or REST), the **software components/“portions”** hosted in the JVM running on the OS of the **service platform server** (*see also* claim element [1.1] above in §IX.B.1 above) *can communicate directly* with the **software components/“portions”** hosted in the JVM running on the OS of the **remote (cloud) server** (*e.g.*, during a secure communication session). *See* my analysis of claim element [1.3] above in §IX.B.1 above).

5. **Claim 14: The system of claim 1, further comprising: a distributed resource service (DRS) which is capable of providing at least one service from a group consisting of: exposing application programming interfaces (APIs) to**

**other applications; configuring and managing platform resources; policy enforcement and authorization of the plurality of network device applications and the plurality of cloud applications to access platform resources; and policy conflict resolution.**

481. Vasell discloses the limitations of claim 14.

482. Vasell discloses a “**cell manager**” (itself a “boxlet ... executing within [a] cell”), managed via the “**management system service**”, that (i) “handles **resource management** so that no single one of the [application-]cells... **consumes too much of the service gateway system** or **service platform server 22** resources, e.g., **CPU time, memory, persistent storage**, and the like”, and (ii) “**manages** operation of the [application-]cells ... so that the right set of the cells ... are running to provide tasks pertaining to the **service applications** ... that are required at a given time” “using information stored in the **cell table** ... which specifies the cells ... that should be running on the **service gateway system.**” EX1004 (Vasell Patent), 20:45-49, 20:23-31, 20:36-43, FIG. 6 (which I annotate below and refer to in my Declaration as “Annotated FIG. 6”):

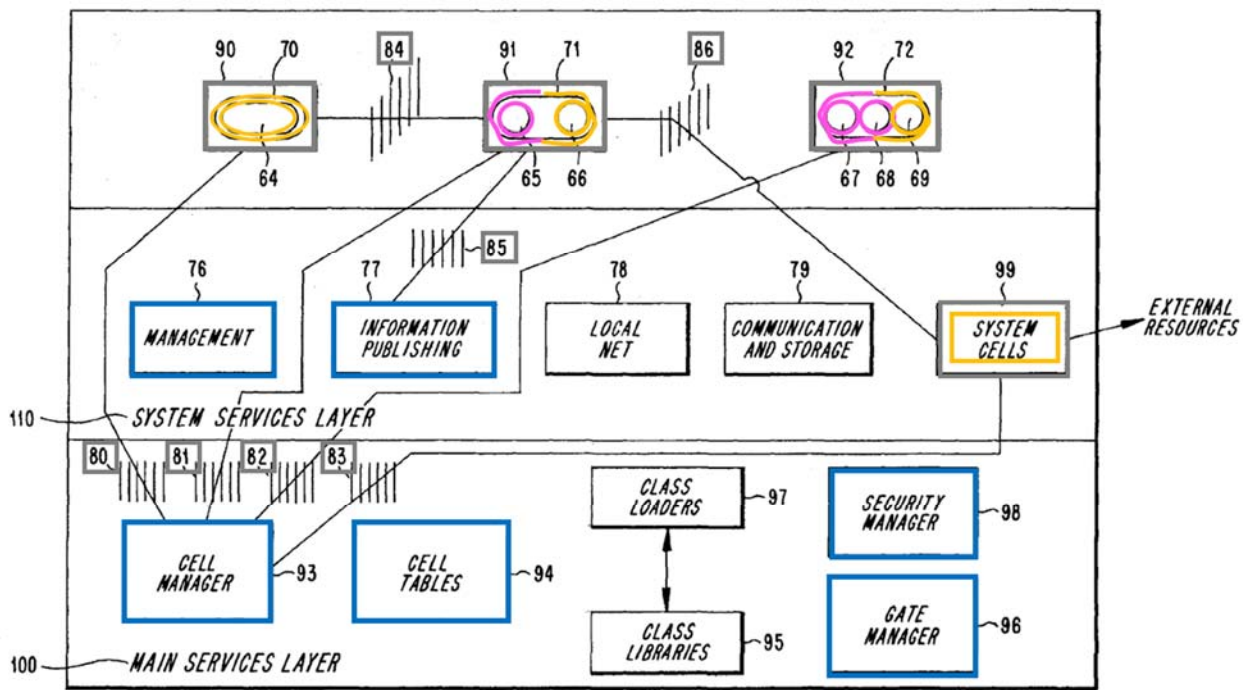


FIG. 6

483. Vasell further discloses that this “cell manager” functionality includes “resource requirements... be[ing] **dynamically allocated based upon a predetermined scheme of maximum allowed or minimum required resources**, for a given situation” so that “different ones of the **service applications**” do not “inadvertently interact in such a way that would cause them to... detrimentally affect each other”, and to “allow[] for resource tradeoffs between the **service applications**... so as to maximize user benefit for a given amount of resources.” *Id.*; EX1004 (Vasell Patent), 12:51-63, 13:12-27.

484. Vasell also discloses that (i) that “[e]ach of the [application-]cells... may be characterized according to its **quota** of resources (e.g., CPU, memory, persistent storage, and the like)”, and (ii) “**diagnostic software**” to “monitor”,

“maintain and control” the distributed **service applications** implementing particular services in the **service gateway system** as part of the “capability and responsibility of the **network operator server 24**” (and of **service provider equipment 34** in embodiments without a **network operator server 24**), which a POSITA would have understood as facilitating, among other things, such **service gateway system** “resource **management.**” *Id.*; EX1004 (Vasell Patent), 14:15-17, 13:28-36, Annotated FIG. 6 (above), 9:20-41, 6:8-20, 8:53-9:18, 14:2-7, 12:27-29, Annotated FIG. 2 (*see my analysis of claim element [1.0] above (§IX.B.1 above)*), Annotated FIG. 5 (*see my analysis of claim element [1.1] above (§IX.B.1 above)*), 11:40-46; *see also my analysis of claim element [1.3] above (§IX.B.1 above)*.

485. Moreover, as I discuss in my analysis of claims 15 (§IX.B.6 below) and 8 (§IX.C.5 below), Vasell discloses the implementation of “listeners” in the **service gateway system** using “proxies” and to send notifications of the occurrence of a specified event between distributed **service application** components and/or between a **service application** and the “**main**” and/or “**system**” “**services layer**” components (*e.g.*, **service application** and **monitoring service application**; **monitoring service application** and **cell manager**, etc.). *See my analysis of claims 11 (§IX.B.6 below), 8 (§IX.C.5 below); EX1004 (Vasell Patent), 13:39-48, 18:16-19, 19:5-41, Annotated FIG. 6 (above), 12:21-29.*

486. Accordingly, a POSITA reading Vasell would understand that Vasell

discloses that, as part of implementing these “resource management”, enforcement of [application-]cell-specific resource “quota[s]”, and “dynamic[] allocat[ion]” of “resource requirements” functions for the plurality of distributed service applications implementing services in the service gateway system, respective instructions would be provided from Vasell’s “management system service” to respective management software (e.g., respective “main” and/or “system” “service boxlet[s]”, “diagnostic software”) distributed between the local and remote (cloud) servers (the service platform server, the network operator server(s), and the service provider equipment) of the service gateway system. *Id.*

487. **In sum**, in my opinion, a POSITA would have understood that, although it uses different words, Vasell’s disclosure of the “cell manager”, managed via the “management system service”, controlling respective, distributed management software (“service boxlet[s]”, “diagnostic software”), to enforce [application-]cell-specific resource “quota[s]”, “handle[] resource management” and “dynamically allocate[]” “resource requirements... based upon a predetermined scheme of maximum allowed or minimum required resources, for a given situation” provides an equivalent disclosure to the ’823 Patent’s description of *the system further comprising a distributed resource service (DRS) which is capable of providing at least one service from a group consisting of... configuring and managing platform resources*. *Id.*; see also the Summary of the ’823 Patent §VII.A

above, (¶¶265-267 above, 270 above) (*citing* EX1001 ('823 Patent), *e.g.*, 21:7-47, FIG. 16, Table 8 (Use cases #20, #67)).

**6. Claim 15: The system of claim 1 further comprising: a distributed content service wherein contents generated by the at least one of the plurality of network device applications are shared with at least one of the plurality of cloud applications.**

488. Vasell discloses the limitations of claim 15.

489. Specifically, a POSITA would have understood that part of the secure communications between the distributed service application “portions”/components would include generated content, such as, for implementation of a “freight management service”, “client software package... on the service platform server 22 [may] provide freight company FMS #1 with information regarding the current location of the truck and its anticipated arrival time of delivery” [*distributed content service wherein contents generated by the at least one of the plurality of network device applications are shared with at least one of the plurality of cloud applications*]. EX1004 (Vasell Patent), Annotated FIG. 5 (*see also* my analysis of claim element [1.1] above (§IX.B.1 above)), 11:13-29; *see also* EX1004 (Vasell Patent), Annotated FIG. 2 (*see also* my analysis of claim element [1.0] above (§IX.B.1 above)), 8:53-9:3, 9:13-24 (Vasell disclosing, where a “portion of the software used to implement” an “environmental control service” resides on service platform server 22, another “portion of the software” residing on network operator

server 24 or on service provider equipment 34) may “turn[] on an air conditioning unit or a heater in connection with” such service).

490. Moreover, Vasell discloses (i) “communication and storage system service” of “system services layer 110” “provides persistent storage that the service applications... can use for communicating among themselves”, and (ii) “cell manager” and “gate manager” control the creation, implementation, and closing of “gates” and “proxies” between distributed service application “portions”/components running on respective programmable network devices (see my analysis of claim elements [1.1] above (§IX.B.1 above), [1.3]-[1.4] above (§IX.B.1 above)) and programmable cloud devices (see my analysis of claim elements [1.2]-[1.4] above (§IX.B.1 above)) in the service gateway system, which is recorded in the “cell table” and “import/export tables within the main services layer”, and through which “arguments and return values”, “nonserialized objects” and “input/output data streams” [*contents generated by the at least one of the plurality of network device applications*] are shared between respective service-application-specific-cells “without compromising integrity of the cells” [*are shared with at least one of the plurality of cloud applications*]. EX1004 (Vasell Patent), 22:49-53, 18:1-30, 18:56-19:41, 20:20-31, 12:21-29, Annotated FIG. 2 (see my analysis of claim element [1.0] above (§IX.B.1 above)), Annotated FIG. 6 (see my analysis of claim 14 (§IX.B.5 above)).

491. **In sum**, in my opinion, a POSITA would have understood that, although it uses different words, Vasell’s disclosure of controlled implementation of “gates” and “proxies” in the **service gateway system**, using “**communication and storage system service**”, to securely and “transparent[ly]” share generated contents between the respective underlying Java software “**portions**”/**components** of Vasell’s distributed **service applications**, provides an equivalent disclosure to the ’823 Patent’s description of *a distributed content service wherein contents generated by the at least one of the plurality of network device applications are shared with at least one of the plurality of cloud applications. Id.; see* my analysis of claim element [1.3] (§IX.B.1 above); *see also* the §VII.A above (Summary of the ’823 Patent) (§§269, 270 above) (*citing* EX1001 (’823 Patent), 21:49-60, Table 8 (Use cases #5, #20, #38-#40, #51, #61-#62, #64)); *see also* my analysis of claim 14 (§IX.B.5 above).

## 7. Claim 19

492. Claim 19 is a system claim corresponding to several of the elements recited in claim 1 above, and, in my opinion, each limitation recited in Claim 19 is invalid for the same reasons discussed regarding Claim 1 above (§IX.B.1 above) and the additional reasons discussed below.

### **[19.0] A system comprising:**

493. Vasell discloses this limitation for the same reasons that I discussed

regarding claim element [1.0] above in §IX.B.1 above.

***[19.1] a programmable network device adapted to host a plurality of network device applications;***

494. Vasell discloses this limitation for same the reasons that I discussed regarding claim element [1.1] above in §IX.B.1 above.

***[19.2] a programmable cloud device adapted to host a plurality of cloud applications***

495. Vasell alone, in view of Alves, or in view of Alves and Rellermeyer, discloses this limitation for the same reasons that I discussed regarding claim element [1.2] above in §IX.B.1 above.

***[19.3] wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications;***

496. Vasell alone, or in view of Alves, discloses this limitation for the same reasons that I discussed above regarding claim element [1.3] above in §IX.B.1 above.

***[19.4] wherein the plurality of network device applications and plurality of cloud applications device form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware.***

497. Vasell discloses this limitation for the same reasons that I discussed regarding claim element [1.4] in §IX.B.1 above.

*[19.5] an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications; and*

498. Vasell discloses this limitation.

499. Vasell discloses that the **service gateway system** is “remotely managed”, for example, via “**management system service 76**” of “system services layer 110”, which “is “implemented in the form of the boxlets[] executing within cells”, and that “provides **an external interface** through which the **service applications** ... may be **downloaded, installed, removed,** executed, and controlled”, which are management operations that Vasell describes, in detail, as being performed by a “**cell manager**” using a “**cell table**” and “facilitate **management of the service applications... lifetimes**” [*application management portal managing life cycles of the plurality of network device applications and plurality of cloud applications*]:

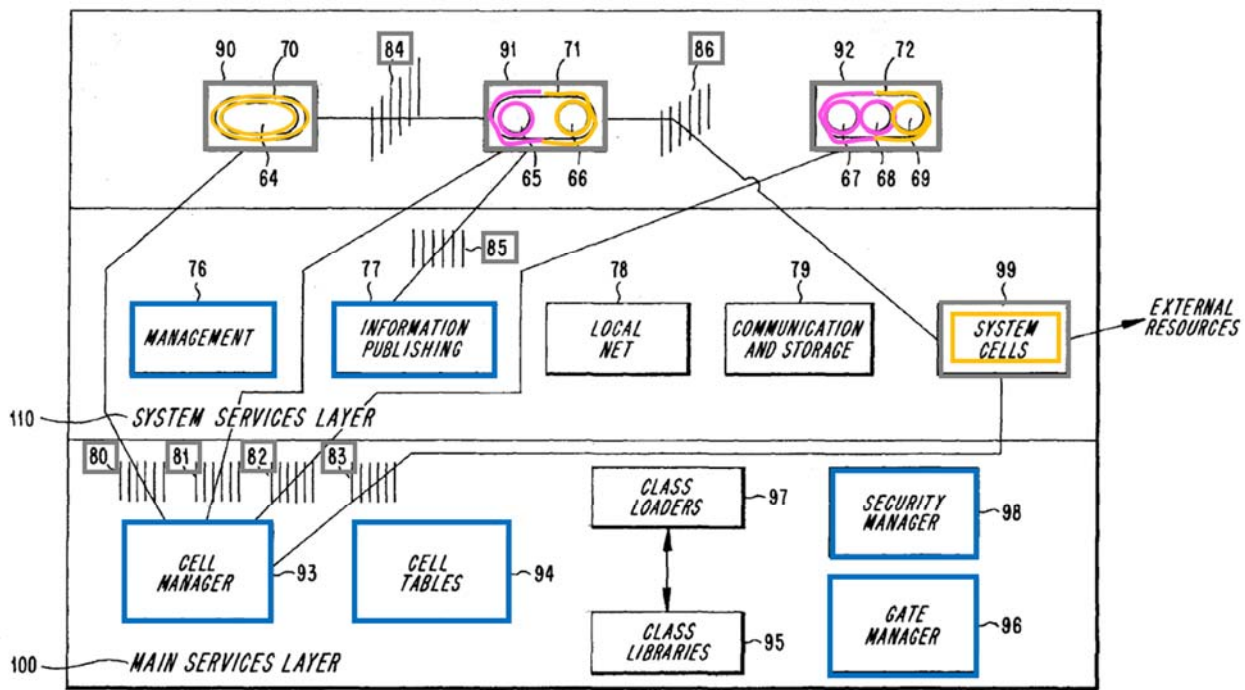


FIG. 6

EX1004 (Vasell Patent), FIG. 6 (which I annotate above and refer to in my Declaration as “Annotated FIG. 6”), 12:11-29, 20:23-31, 22:34-37, 20:49-53, 20:10-22:30, 15:58-17:39, 4:44-50, 5:16-26.

500. Vasell 2<sup>nd</sup> Provisional provides details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 7-8 (“On the *e-box* service platform, applications that implement services can be **downloaded, installed or removed**. The [service] platform also permits the **remote life-cycle management** of service applications. ... The platform consists of two layers (Figure 4): • Main services, which control applications. • System services, which offer utilities to the service applications. ... When executed as a service application, the boxlet code is executed within a cell ... Cells are created and controlled by the cell manager, which is part of the main services layer (Figure

4). All cells are registered in a **cell table**.”), Figure 4 (“Organization of the service platform”, including “**Information publishing (Web, WAP, etc.)**”, “**cell manager**”, “**cell table**”), 8 (description of “*System services*” and “*Service application management*: The **management** of **service applications**—**downloading, installation**, execution, control, and **removal**—is performed via a **specific O&M interface** implemented as a **system service boxlet**.”), 9 (“Network operator... operates and maintains the system service registry ... The NO sells network access to service providers and can have roaming agreements with other NOs to provide a global service network.”).

501. Vasell provides further details on these “remote life-cycle **management** of **service applications**” operations, including, for example, that the “**cell manager**” (i) “is responsible for starting or activating [application-]cells... as well as and stopping and destroying [application-]cells...”, (ii) “stor[es] the state of the **service applications** ... in the **cell table** ... to boot the **service applications** ... of the cells ... in any order”, and (iii) “manag[es] the creation and destruction of the [application-]cells”, on the respective **servers** in the **service gateway system**. EX1004 (Vasell Patent), 20:32-34, 20:40-43, 21:13-26, Annotated FIG. 6 (above).

502. **In sum**, Vasell discloses an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications (Vasell’s **management system service** includes an “external

interface” through which the lifecycle operations of the plurality of distributed, Java **service applications** in the **service gateway system** are **remotely managed** including, for example, the downloading, installing, starting, activating, stopping, destroying, and removing of the distributed **service applications** implementing particular services in the **service gateway system**).

*[19.6] an infrastructure application marketplace in communication with the application management portal, said infrastructure application marketplace capable of providing tested distributed applications to the application management portal.*

503. Vasell in view of Alves discloses this limitation.

504. As I explained in my analysis of claim element [19.5] above (§IX.B.7 above), Vasell discloses **management services** that, in part, provide “an **external interface** through which **service applications**... may be downloaded, installed...” [*application management portal* of [19.5] above (§IX.B.7 above)]. See my analysis of claim element [19.5] above (§IX.B.7 above) (*citing, e.g.*, EX1004 (Vasell Patent), 22:34-37, 13:37-43, 20:45-54; EX1016 (Vasell 2<sup>nd</sup> Provisional), 4, 7-9).

505. Vasell discloses that “[s]ervice gateways according to the present invention are preferably **open** platforms to allow for the development of **service applications** by **third party developers**” using “standard communication protocols and programming languages”, “service gateways according to the present invention are intended to be horizontal in nature so that a plurality of service providers can

share the same platform infrastructure”, and “service gateway compatibility potentially opens **third-party markets** for... services.” EX1004 (Vasell Patent), 3:10-19, 5:30-33, 10:18-23, 2:60-66 (“[t]he **service gateway system** according to the present invention facilitates the development... of services... [by] **different services providers**” where “[e]ach service... [is] implemented as software **service applications**.”), 13:37-43.

506. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“*Development environment for **application software***: The development environment must follow the ‘write-once, run-everywhere’ maxim and should be based on Java standards. New **applications** will interact with the **e-service infrastructure** through Java application program interfaces (API) that comply with mainstream Java development. By leveraging the Java development, the **application software** environment can be taken to a higher level of abstraction, allowing nonspecialists to develop **service applications** more easily.”), 8 (“*Development environment*: Boxlets are created using a standard Java development environment. For instance, the Java development kit (JDK) from Sun Microsystems can be used as well as other development environments. The only parts that are specific to boxlet development are the libraries that contain APIs for the main services and system services layers.”), 9 (identifying the disclosed “new **e-service infrastructure**” as “an open platform that can be used by several independent service providers”).

507. Although Vasell also identifies that “**service provider equipment 34...** **may** be the initial repository for **software** used to implement connectivity based services” of such a service provider, and that “**network operator server 24** may need to **receive some portions** of the **service application software** from the service provider” to then “**download some ... of that [received] software** to the **service platform server 22**” [*distributed applications to the application management portal*], Vasell does not explicitly disclose a **centralized** application repository storing the respective **service applications** developed by each of the “**different** service providers”, “**third party** developers” and “nonspecialists” in this intended “open” development environment with “third party markets for... services.” *Id.*; EX1004 (Vasell Patent), 7:14-18, 8:29-36, 8:47-9:18.

508. In my opinion, as I discussed in the Motivation to Combine §IX.A above, it would have been obvious to modify Vasell’s teachings with Alves’s teachings of implementing a **centralized OSGi repository**. See §IX.A above (Motivation to Combine) (*citing* §VIII.A.2 above (Overview of the Prior Art – Alves) *citing* EX1008 (Alves), 126-130, 263, 266-267, FIG. 10.10 (which I reproduced and annotated in §VI.F.1 above), FIG. 10.12 (which I reproduced and annotated in §VIII.A.2 above).

509. For example, as I discussed in the Motivation to Combine §IX.A above, Alves discloses, in a private/community/hybrid cloud OSGi implementation, where

the OSGi remote management services (rather than a cloud provider) continues to “manage OSGi framework instances and their bundles” (e.g., FIGS. 10.10, 10.12 (reproduced and annotated below, *see also* my analysis of claim element [1.2] above (§IX.B.1 above))), (i) a “cloud provider could give [the management services] access to an OSGi bundle repository (OBR)” in the provider’s cloud, from which management services “could install, update, and uninstall bundles as needed”, and for which developers “can design and validate [their] applications using any OSGi implementation in a standalone environment before moving [them] to the cloud”, and, (ii) for any bundle updates, the software developer would “just need to inform the cloud provider of the update”, and “OSGi would let [the management services] update [the] system quite easily and efficiently”, because “OSGi application[s] already hav[e] to deal with the fact that dynamic changes and updates may happen, so the fact that it’s happening when it’s running in the cloud or running as a standalone local application is mostly immaterial.”

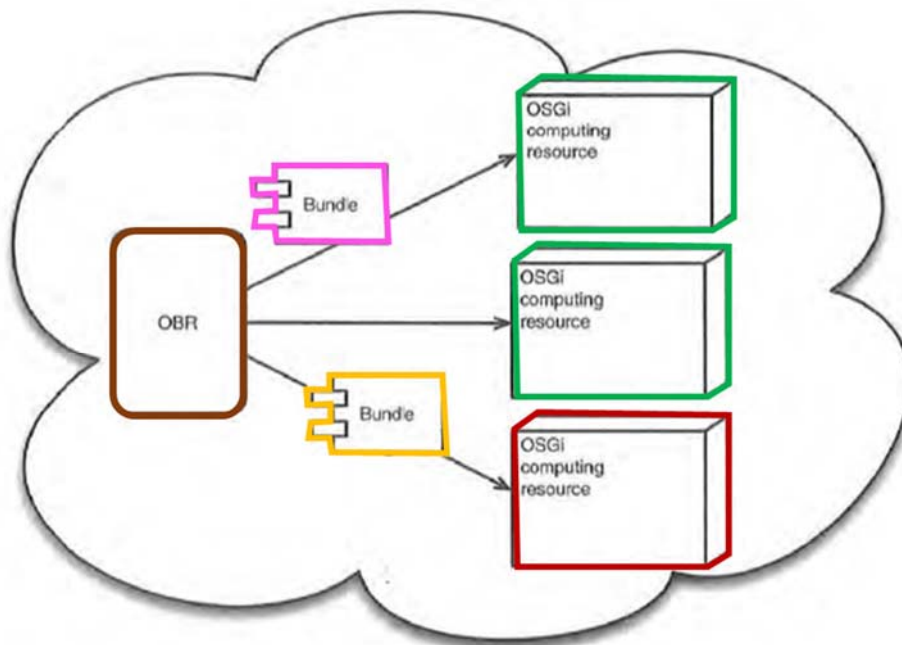


Figure 10.12 OBR and OSGi framework as a platform for cloud computing

*Id.* (citing EX1008 (Alves), 263, 266-267 (bold emphasis added), FIG. 10.12 (annotated)).

510. Additionally, as I discussed in the Motivation to Combine §IX.A above, Alves discloses the importance of software developers “validat[ing]” (testing and certifying as successfully completing such testing) their applications, e.g., before “moving [them] to the cloud” OBR, and technical implementation details for “effectively” and “eas[ily]” doing such application testing, including that application developers will “improve robustness by testing your applications” both (i) “at the unit level ... to make sure that the bundle ... works in isolation”, and (ii) using a “process ... called *integration testing*” “to make sure the bundles are able to talk to each other.” *Id.* (citing EX1008 (Alves), 266, 126-130 (italicized emphasis in the

original)).

511. Accordingly, as I discussed in the Motivation to Combine §IX.A above, a POSITA would have been motivated to modify Vasell’s teachings with Alves’s teachings of having a cloud provider give Vasell’s **service gateway system management services** (see my analysis of claim element [19.5] above (§IX.B.7 above)) access to an **application store**, as was well-known in the art (see Technology Background §VI.F.1 above (Distributed Systems – OSGi) (¶¶200-203 above); Technology Background §VI.F.2 above (Distributed Systems – Clouds) (¶¶224-226 above)), in which “validate[d]” (tested and certified as successfully tested) **service applications** (and upgrades of the same) would be stored.

512. Specifically, in this example, Vasell’s **management services** that, in part, provide “an **external interface** through which **service applications**... may be downloaded, installed...”, continue to **manage** the **service gateway system** servers (see my analysis of claim elements [1.1]-[1.2] above in §IX.B.1 above) via Vasell’s “**management system service**”, “**cell manager**”, “**cell table**”, (see my analysis of claim element [19.5] above in §IX.B.7 above) [*application management portal*]. Vasell’s **management services** could, as needed, download such “validate[d]” **service applications** (and upgrades of the same) from such **cloud-provider-application-store** for installation in the **local** and **remote (cloud)** servers of such system [*an infrastructure application marketplace in communication with the*

*application management portal, said infrastructure application marketplace capable of providing tested distributed applications to the application management portal]* as a mere application of a known solution to a known **distributed system** ready for improvement to achieve known and predictable results. *Id.*; see §IX.A above (*citing* §VIII.A.2 above (Overview of the Prior Art – Alves) *citing* EX1008 (Alves), 126-130, 263, 266-267, FIG. 10.10 (which I reproduced and annotated in §VI.F.1 above), FIG. 10.12 (which I reproduced and annotated in §VIII.A.2 above)).

513. **In sum**, in my opinion, Vasell’s teachings in view of Alves’s teachings discloses an infrastructure application marketplace in communication with the application management portal (cloud provider’s **OSGi bundle repository (OBR)** in communication with Vasell’s **management system service**, which includes an “external interface” through which the lifecycle operations of the plurality of distributed, Java **service applications** in the **service gateway system** are **remotely managed**) capable of providing tested and certified distributed applications to the application management portal (capable of providing “validate[d]” (tested and certified as successfully tested) Java **service applications** (and upgrades of the same) to the **management services**, which could, as needed, download such **applications** (and upgrades) for installation in the **local** and **remote (cloud)** servers of the **service gateway system**).

**C. Claim Element by Claim Element Invalidity Analysis: '823 Patent Claims 3-5, 7-8, and 18 are Obvious Over Vasell in View of Alves, Rellermeyer and Hall.**

**1. Claim 3:**

*[3.0] The system of claim 1 further comprising:*

514. Vasell discloses this limitation for the same reasons that I discussed regarding claim element [1.0] above in §IX.B.1 above.

*[3.1] an application management portal capable of managing life cycles of the distributed applications; and*

515. Vasell in view of Alves discloses this limitation for the same reasons that I discussed regarding claim element [19.5] above (§IX.B.7 above).

*[3.2] an infrastructure application marketplace in communication with the application management portal, said infrastructure marketplace capable of providing tested and certified distributed applications to the application management portal.*

516. Vasell in view of Alves, or Vasell in view of Alves and Hall, discloses this limitation.

517. As I explained in my analysis of claim element [19.6] above in §IX.B.7 above, and in the Motivation to Combine §IX.A above, Vasell in view of Alves discloses an infrastructure application marketplace in communication with the application management portal (cloud provider's **OSGi bundle repository (OBR)** in communication with Vasell's **management system service**, which includes an "external interface" through which the lifecycle operations of the plurality of

distributed, Java **service applications** in the **service gateway system** are **remotely managed**) capable of providing tested and certified distributed applications to the application management portal (capable of providing “validate[d]” (tested and certified as successfully tested) Java **service applications** (and upgrades of the same) to the **management services**, which could, as needed, download such **applications** (and upgrades) for installation in the **local** and **remote (cloud)** servers of the **service gateway system**).

518. I note that, as I discussed in §VII.A above (Summary of the ’823 Patent), other than the claims, the ’823 Patent only uses the word “certified” in two places: at (i) 10:50-55 (“[u]sing **fxManager** 306, the admin may discover new fxApps from the **fxStore** 308, which presents all the tested and certified vApps...”), and (ii) 15:14-15, 15:39-49 (stating “[t]he major functions of the **fxManager** 306 may include ... [b]rowsing through the certified applications in the **fxStore**...”).

519. In my opinion, as I discussed in the §IX.A above (Motivation to Combine) and in my analysis of claim element [19.6] above in §IX.B.7 above, a POSITA would have understood that, as Vasell in view of Alves discloses the importance of successful “unit level” and “integration” testing of Java **service applications**, and that such **applications** are “design[ed] **and validate[d]** ... using any OSGi implementation in a standalone environment before moving to the cloud” provider’s **OBR**, Vasell in view of Alves discloses an *infrastructure marketplace*

*capable of providing tested and certified distributed applications to the application management portal.*

520. **In sum**, in my opinion, a POSITA would have understood that Vasell's teachings, in view of Alves's teachings, discloses an infrastructure application marketplace in communication with the application management portal (cloud provider's **OSGi bundle repository (OBR)** in communication with Vasell's **management system service**, which includes an "external interface" through which the lifecycle operations of the plurality of distributed, Java **service applications** in the **service gateway system** are **remotely managed**) capable of providing tested and certified distributed applications to the application management portal (capable of providing "validate[d]" (tested and certified as successfully tested) Java **service applications** to the **management services**, which could, as needed, download such **applications** for installation in the **local** and **remote (cloud)** servers of the **service gateway system**).

521. To the extent that the Patent Owner asserts that Vasell in view of Alves does not disclose "certified" applications in the cloud provider's **OBR**, in my opinion and as I discussed in the Motivation to Combine §IX.A above, a POSITA would have been motivated to modify the teachings of Vasell in view of Alves with Hall's teachings of requiring that OSGi-application-developers use well-known digital signature, key exchange and certificate techniques to "digitally sign" their

respectively developed distributed **application components** (**bundles** and/or application deployment packages including such **bundles**), whereby a “well-known (trusted)” party certifies a respective **application’s** authenticity if **it** is digitally signed by the respective unique private key issued to the accompanying certificate-identified developer. *See* §IX.A above (*citing* §VIII.A.3 above (Overview of the Prior art in the Grounds for Challenge – Hall) (*citing* EX1009 (Hall), 179, 331-334, 457-463, 472-476))).

522. Vasell emphasizes that, due to the “open” nature of the **service gateway system** (*see* claim element [19.6] above in §IX.B.7 above), “mechanisms for **authentication**... are integrated within the **service gateway system** design”, and “various **authorization and identity verification techniques**, etc. [are] to be provided so that the implementation... associated with these [connectivity-based] services is robustly defended against intrusion and manipulation.” EX1004 (Vasell Patent), 4:52-65, 11:59-12:10, 15:46-47.

523. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 3 (“Consumer Requirements of E-Services: ... *Security*: Consumers do not tolerate invasion of privacy or hacking into mission-critical services by strangers or neighbors.”), 4 (“Service Provider Requirements of an **E-Service Infrastructure**: ... *Integrity*: Different service providers must be allowed to share the infrastructure without adversely affecting each other’s services.”), 6 (“[T]he fact that the system

**will** be shared by multiple service providers, make security a major issue. Mechanisms for **authentication and authorization must** be well integrated into the *e-box* system design.”).

524. Moreover, as I discussed in the Motivation to Combine §IX.A above, a POSITA would have been confident that the combination of the teachings of Vasell, Alves and Hall — to require digitally signed Java **service applications** in the **cloud-provider OBR/store** — would have a high likelihood of success, and a reasonable expectation of achieving the increased security benefits expressly disclosed by Vasell (*see* ¶¶522-523 above) and other well-known benefits including, for example, rendering each application “tamperproof”, authenticating the developer of such applications (by digital signature matching, which certifies the signer used the unique private key issued by the “well-known (trusted)” party identified on the X.509 certificate (with the unique public key of the key pair)), allowing Vasell’s **management services** to grant Vasell’s application-cell-specific-access-levels to each such **application** based on the respective developer’s identity (signature, as certified by the “well-known (trusted)” party), etc. *Id.*; §IX.A above (*citing* §VIII.A.3 above (Overview of the Prior art in the Grounds for Challenge – Hall) (*citing* EX1009 (Hall), 179, 331-334, 457-463, 472-476))).

525. **In sum**, in my opinion, to the extent that the Patent Owner asserts that Vasell in view of Alves does not disclose “certified” applications in the cloud

provider's **OBR**, a POSITA would have understood that Vasell's teachings, in view of Alves' and Hall's teachings, discloses an infrastructure application marketplace in communication with the application management portal (cloud provider's **OSGi bundle repository (OBR)** in communication with Vasell's **management system service**, which includes an "external interface" through which the lifecycle operations of the plurality of distributed, Java **service applications** in the **service gateway system** are **remotely managed**) capable of providing tested and certified distributed applications to the application management portal (capable of providing "validate[d]" (tested and certified as successfully tested), and "digitally sign[ed]" (certified to be the developer by the "well-known (trusted)" party identified on the accompanying X.509 certificate), Java **service applications** to the **management services**, which could, as needed, download such **applications** for installation in the **local** and **remote (cloud)** servers of the **service gateway system**).

2. **Claim 4: The system of claim 3, wherein the application management portal is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications.**

526. Vasell discloses this limitation.

527. As I explained in my analysis of claim element [19.5] above in §IX.B.7 above, Vasell discloses the "**cell manager**", using the "**cell table**" and via the

“management system service” including an “external interface” [*application management portal*] manages the *life cycle* operations of the plurality of distributed, Java service applications in the service gateway system, including, for example, the downloading, installing, starting, and activating [*provisioning*] operations, and the stopping, destroying, and removing [*deprovisioning*] operations, of the distributed service applications implementing particular services in the service gateway system.

528. Specifically, as I explained in my analysis of claim element [19.5] above in §IX.B.7 above, Vasell discloses that these same management components “provide[] an external interface through which the service applications... may be **downloaded, installed, removed**, executed, and controlled”, “is responsible for **starting or activating** [application-]cells... as well as and **stopping and destroying** [application-]cells...”, “stor[e] the state of the service applications... to boot the service applications... of the cells... in any order”, and “manag[e] the **creation and destruction** of the [application-]cells”, on the respective devices in the service gateway system. [*wherein the application management portal is capable of ... provisioning said distributed applications; and deprovisioning said distributed applications*]. EX1004 (Vasell Patent), 22:35-38, 20:32-34, 20:40-43, 21:13-26 (example disclosures of provisioning and deprovisioning emphasized in bold); Annotated FIG. 6 (*see* claim element [19.5] above in §IX.B.7 above).

529. **In sum**, in my opinion, Vasell discloses that its application

management portal (Vasell's **management system service** includes an “**external interface**” through which the lifecycle operations of the plurality of distributed Java **service applications** in the **service gateway system** are **remotely managed**) is capable of provisioning said distributed applications and deprovisioning said distributed applications (**remotely manages** the downloading, installing, starting, activating, stopping, destroying, and removing of the distributed **service applications** implementing particular services in the **service gateway system**).

**3. Claim 5: The system of claim 3, further comprising: a distributed resource service (DRS) which controls access to a plurality of resources upon receiving instructions from the application management portal.**

530. Vasell discloses the limitations of claim 5 to POSITAs.

531. As I explained in my analysis of claim 14 in §IX.B.5 above, Vasell discloses a “**cell manager**”, managed via the “**management system service**” including an “**external interface**” [*application management portal*], controlling and providing instructions to respective **management software** (e.g., respective “main” and/or “system” “service boxlet[s]”, “diagnostic software”) distributed between the **local** and **remote** servers of the **service gateway system**, to “**handle[] resource management**” “so that no single one of the [application-]cells... **consumes too much of the service gateway system ... resources**, e.g., **CPU time, memory, persistent storage**, and the like”, enforce [application-]cell-specific **resource “quota[s]”, and “dynamically allocate[]” “resource requirements... based upon**

a predetermined scheme of maximum allowed or minimum required resources, for a given situation” [*a distributed resource service (DRS) which controls access to a plurality of resources upon receiving instructions from the application management portal*]. See my analysis of claim 14 in §IX.B.5 above.

532. As I also explained in my analysis of claim element [1.3] above in §IX.B.1 above, consistent with the well-known sandboxing of the run-time environment of JVMs, Vasell discloses that each of the underlying **software modules** that collectively form each of the distributed **service applications** is “designated into various categories having different degrees of access restriction”, where “access rules” (**administrator-defined** for each **application**’s “cell” (which “represent[s] the resources available to” the particular **application**) and for application cell “groups”) are controlled and enforced using, in part, “inter-cell communication... controlled interfaces” called “gates” (managed by “**gate manager 96**”), including to “limit[], monitor[], and control[]” “access of the cells.” See my analysis of claim element [1.3] above in §IX.B.1 above; see also EX1004 (Vasell Patent), Annotated FIG. 6, 12:19-29, 13:28-36 (“**service applications** 70-72 include the boxlets 64-69 within the cells 90-92” and that “[c]ells 90-92 represent the resources available to the **service applications** 70-72”), 14:2-7.

533. Vasell’s 2<sup>nd</sup> Provisional provides additional details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 8 (description of “*service applications*”, “The management of

service applications— ..., execution, control, ... -is performed via a specific O&M interface implemented as a system service boxlet. This system service is primarily used by the management system.”).

534. In other words, a POSITA would understand that Vasell discloses that “access rules” (administrator-defined for each application’s “cell” (which “represent[s] the resources available to” the particular application) and for application cell “groups”) are controlled and enforced using, in part, “inter-cell communication... controlled interfaces” called “gates” (managed by “gate manager 96”), including to limit, monitor, and control the “resources (e.g., CPU, memory, persistent storage, and the like)” that are respectively available, at any given point in time, to each of the distributed service application software components hosted in respective JVMs running on the respective OSs of the local and remote (cloud) servers of the service gateway system. *Id.*

535. Moreover, as I discuss in my analysis of claim 14 in §IX.B.5 above, Vasell discloses the implementation of “listeners” in the service gateway system using “proxies” and to send notifications of the occurrence of a specified event between distributed service application modules and/or between a service application and the “main” and/or “system” “services layer” components (e.g., service application and monitoring service application; monitoring service application and cell manager, etc.). *See* my analysis of claims 14 in §IX.B.5 above;

EX1004 (Vasell Patent), 13:39-48, 18:16-19, 19:5-41, Annotated FIG. 6, 12:21-29.

536. As an example of Vasell's **distributed management software** [*distributed resource service*], upon receiving instructions from Vasell's "management system service" having an "external interface through which the **service applications** ... may be ... executed, and controlled" [*from the application management portal, see claim element [3.1] above (§IX.C.1 above)*]), controlling access to a plurality of **service gateway system resources** by the underlying software components that collectively form at least two of the distributed **service applications** (*see claim element [1.3] above (§IX.B.1 above)*), Vasell discloses:

To achieve a secure, robust environment, different ones of the **service applications** 70-72 are not allowed to inadvertently interact in such a way that would cause them to fail or detrimentally affect each other. For example ... [a]n ... embodiment allows for resource tradeoffs between the **service applications** 70-72, so as to maximize user benefit for a given amount of resources. For example, it may be the case that two of the **service applications** 70-71, e.g., a **video telephone service application** and a **3-D interactive video game**, are very resource intensive and cannot be used at the same time. In such a situation, when one of the resource intensive **service applications** 70-71 is being used and someone seeks to start another (e.g. make a video telephone call while the 3-D game is running), a user prompt may be provided according to a predetermined scheme. The user prompt may, for instance, give the option of using the video telephone at lower picture

quality, or inform the user that making the video telephone call will result in lower quality rendering of the 3-game background images.

EX1004 (Vasell Patent), 12:51-54, 13:12-27; *see also* 12:27-29 (“the **software elements** of the **service gateway system** may be **distributed** among various units of geographically separated equipment, as shown in FIG. 2”).

537. Vasell 2<sup>nd</sup> Provisional adds more detail. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“Services should be implemented as distributed **applications**. By executing over several **infrastructure nodes**, more complex services can be developed without the need for extremely powerful and complex **edge servers**. Distributed **applications** will considerably increase the technical life and reduce the cost of the **edge servers**.”), 5 (Figure 2), 7-8 (“A **service application** typically fulfills the role of a gateway between **servers** in the external network (from which the service is accessed and controlled) and **devices** in the local network. The **service application** is the central entity, and **its** authorities are defined by the **service platform**” including “**Main services**, which **control** applications” and “**System services**, which offer **utilities** to the **service applications**.”), 9 (Figure 5, “Box D, Distributed Services System Architecture” including, *e.g.*, “The local component of the service is implemented using Java **applications** that run on the **edge server** (and optionally, in the **management system** [“a key tool for the **NO** [**network operator**]”] and at the site of the **SP** [**service provider**].”).

538. A POSITA would have understood that, as part of implementing “resource tradeoffs” for two or more resource-intensive distributed **service applications** in the **service gateway system**, respective instructions would be provided from Vasell’s “**management system service**” to respective **management software** (e.g., respective “main” and/or “system” “**service boxlet[s]**”, “**diagnostic software**”) distributed between the **local** and **remote (cloud)** servers of the **service gateway system**. *Id.*; see also my analysis of claim 14 in §IX.B.5 above.

539. **In sum**, in my opinion, a POSITA would have understood that, although it uses different words, Vasell’s disclosure of a “**cell manager**”, managed via the “**management system service**” having an “**external interface**”, controlling and providing instructions to this respective, **distributed management software** (“**service boxlet[s]**”, “**diagnostic software**”), to implement “resource tradeoffs” for two or more resource-intensive distributed **service applications** in the **service gateway system**, provides an equivalent disclosure to the ’823 Patent’s description of *a distributed resource service (DRS) which controls access to a plurality of resources upon receiving instructions from the application management portal*. *Id.*; see my analysis of claim 14 in §IX.B.5 above; see also the Summary of the ’823 Patent §VII.A above, (¶¶265-267 above, 270 above) (*citing* EX1001 (’823 Patent), e.g., 21:7-47, FIG. 16, Table 8 (Use cases #20, #67)).

4. **Claim 7: The system of claim 5, wherein the DRS further includes a load controller adapted to monitor loads on at least**

**one of the plurality of network device applications and at least one of the plurality of cloud applications and effect change in accordance with thresholds received from the application management portal.**

540. Vasell alone, or in view of Rellermeyer, discloses the limitations of claim 7.

541. As I explained in my analysis of claim 14 (§IX.B.5 above) and claim 5 (§IX.C.3 above), Vasell discloses a “**cell manager**”, managed via the “**management system service**” having an “**external interface**”, controlling and providing instructions to respective, **distributed management software** (“**service boxlet[s]**”, “**diagnostic software**”), to (i) enforce [application-]cell-specific “quota[s] of resources (e.g., CPU, memory, persistent storage, and the like)”, (ii) “**handle[] resource management** so that no single one of the [application-]cells... **consumes too much of the service gateway system ... resources, e.g., CPU time, memory, persistent storage, and the like**”, and (iii) “**monitor**”, “**maintain and control**” the distributed **service applications** implementing particular services in the **service gateway system** [*the DRS further includes a load controller adapted to monitor loads on at least one of the plurality of network device applications and at least one of the plurality of cloud applications*]. See my analysis of claims 14 (§IX.B.5 above) and 5 (§IX.C.3 above).

542. As I also explained in my analysis of claim 14 (§IX.B.5 above), Vasell also discloses the functionality provided by the “**cell manager**”, managed via the

“management system service”, controlling this respective, distributed management software (“service boxlet[s]”, diagnostic software”), includes “resource requirements... be[ing] **dynamically allocated based upon a predetermined scheme of maximum allowed or minimum required resources**, for a given situation” [*the DRS further includes a load controller adapted to ... effect change in accordance with thresholds received from the application management portal*] so that “different ones of the **service applications**” do not “inadvertently interact in such a way that would cause them to... detrimentally affect each other”, and to “allow[] for resource tradeoffs between the **service applications**... so as to maximize user benefit for a given amount of resources.” See my analysis of claim 14 (§IX.B.5 above); EX1004 (Vasell Patent), 12:51-63, 13:12-27.

543. In my opinion, a POSITA would have understood that taking actions based on a minimum or maximum value constitutes acting on those values as thresholds.

544. Moreover, as I discussed in my analysis of claim 14 (§IX.B.5 above), Vasell discloses the implementation of “listeners” in the **service gateway system** using “proxies” and to send notifications of the occurrence of a specified event between distributed **service application** components and/or between a **service application** and the “main” and/or “system” “**services layer**” components (*e.g.*, **service application** and **monitoring service application**; **monitoring service**

application and cell manager, etc.). See my analysis of claim 14 (§IX.B.5 above); EX1004 (Vasell Patent), 13:39-48, 18:16-19, 19:5-41, Annotated FIG. 6, 12:21-29.

545. In my opinion, to the extent that the Patent Owner asserts or the Board finds that this is a means-plus function limitation, Vasell’s “resource management” and “monitor[ing]” functionality to ensure “that no single one of the [application-]cells... consumes too much of the service gateway system ... resources”, and including “resource requirements... be[ing] dynamically allocated based upon a predetermined scheme of maximum allowed or minimum required resources, for a given situation” so that “different ones of the service applications” do not “inadvertently interact in such a way that would cause them to... detrimentally affect each other”, and to “allow[] for resource tradeoffs between the service applications... so as to maximize user benefit for a given amount of resources”, is also the structure identified by the Patent Owner in the specification of the ’823 Patent for “a load controller” performing the functions of this claim 7 (§IX.C.4), or an equivalent thereof. *Id.*

546. As identified in §VIII.B.2 above (Claim Construction), in its Claim Construction brief in the Related Litigation, the Patent Owner identified “a controller executing at least one of a ‘cloud breathing’ or ‘load monitoring’ application”, as described at EX1001 (’823 Patent), 21:66-22:5, 22:20-22 (load monitoring application), 15:64, 22:9-26, FIGS. 17A-17B (cloud breathing application), as the

structure corresponding to the recited functions of this claim 7 (§IX.C.4) and the portions of the specification describing this structure. *See* §VIII.B.2 above (Claim Construction) (*citing* EX1070 (Patent Owner Claim Construction Brief), 16-19; EX1071 (Petitioner Claim Construction Brief), 19-20).

547. Moreover, as I discussed in this claim 7 (§IX.C.4 above) the “way” that Vasell discloses its “resource management” and “monitor[ing]” functionality “monitor[ing] loads ... and affect[ing] change in accordance with thresholds received” is that Vasell’s “resource management” and “monitor[ing]” functionality ensuring “that no single one of the [application-]cells... consumes too much of the service gateway system ... resources”, and including “resource requirements... be[ing] dynamically allocated based upon a predetermined scheme of maximum allowed or minimum required resources, for a given situation” so that “different ones of the service applications” do not “inadvertently interact in such a way that would cause them to... detrimentally affect each other”, and to “allow[] for resource tradeoffs between the service applications... so as to maximize user benefit for a given amount of resources.”

548. In my opinion, a POSITA would have understood that Vasell’s disclosed “way” of performing the recited functions of this claim 7 (§IX.C.4 above) is substantially similar (at least by being one example thereof) of the “way” that a “load controller adapted to monitor loads ... and affect change in accordance with

thresholds received ...” is described in EX1001 (’823 Patent), 21:66-22:5, 22:20-22 (load monitoring application), 15:64, 22:9-26, FIGS. 17A-17B (cloud breathing application). *Id.*; *see also* §VIII.B.2 above (Claim Construction).

549. Also, in my opinion, a POSITA would have understood that the “result” of Vasell’s disclosure of its “resource management” and “monitor[ing]” functionality ensuring “that no single one of the [application-]cells... consumes too much of the service gateway system ... resources”, and including “resource requirements... be[ing] dynamically allocated based upon a predetermined scheme of maximum allowed or minimum required resources, for a given situation” so that “different ones of the service applications” do not “inadvertently interact in such a way that would cause them to... detrimentally affect each other”, and to “allow[] for resource tradeoffs between the service applications... so as to maximize user benefit for a given amount of resources”, is also substantially similar to the “result” in the ’823 Patent’s disclosure. *Id.*

550. Accordingly, in my opinion, a POSITA would have understood that the structure in Vasell for performing the function of “monitor[ing] loads ... and affect[ing] change in accordance with thresholds received” is at least equivalent to the Patent Owner identified structure in the specification of the ’823 Patent. *Id.*

551. To the extent that the Patent Owner asserts that this claim requires monitoring and control functionality for load (resource usage) expansion or

contraction across multiple servers in Vasell’s **service gateway system** and that Vasell does not explicitly disclose this, it would have been obvious to modify Vasell’s teachings with Rellermeyer’s (EX1011) teachings of having a **topology manager**, (i) upon receiving notification of an “**overloaded**” **cloud server**, “re-bind a **client** to a different” **cloud server** to “re-distribut[e] the load among” the **cloud servers**, and (ii) upon receiving notification of “service **over-utilization**” on a **cloud server**, “start a new instance of the service” on a different **cloud server** and “bind some of the **clients** to the new service” instance, as I discussed in my analysis of claim element [1.2] above (§IX.B.1 above), and in the Motivation to Combine §IX.A above. *See* my analysis of claim element [1.2] above (§IX.B.1 above); §IX.A above (Motivation to Combine *citing* §VIII.A.4 above (Overview of Rellermeyer) *citing* EX1011 (Rellermeyer), 2-5).

552. Vasell 2<sup>nd</sup> Provisional provides additional details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 9 (“The [Network Operator] sells network access to service providers and can have roaming agreements with other NOs to provide a global service network.”).

553. In my opinion, as I discuss in the Motivation to Combine §IX.A above, a POSITA would have been confident that, given Rellermeyer’s express disclosure that its solution “uses features already provided by OSGi” and, as I discussed in my analysis of claim element [1.2] above (§IX.B.1 above), given that a POSITA would

have understood that Vasell’s disclosed features of its **service gateway system** (for which technical implementation details were later standardized in the open OSGi specifications) made it an “ideal” system to implement in a cloud environment, this modification had a reasonable likelihood of achieving known and predictable benefits, including, for example, turning Vasell’s connectivity-based services “into elastic units of deployment which can be migrated and replicated across multiple cloud nodes to increase the availability and reliability of the administered service.”

*Id.*

554. Moreover, for the same reasons as discussed in the Motivation to Combine §IX.A above, a POSITA would have been motivated to modify Vasell’s teachings with Rellermeyer’s teachings of load balancing and elastic scalability, which, in my opinion, discloses the “controller executing... a ‘cloud breathing’ **[and]** ‘load monitoring’ application” indicated in the ’823 Patent specification (*see* §VIII.B.2 above) or equivalent thereof. *See* §VIII.B.2 above (*citing* EX1001 (’823 Patent), 21:66-22:5, 22:20-22 (load monitoring application), 15:64, 22:9-26, FIGS. 17A-17B (cloud breathing application), as the structure corresponding to the recited functions and the portions of the specification describing this structure.) (*citing* EX1070 (Patent Owner Claim Construction Brief), 16-19; EX1071 (Petitioner Claim Construction Brief), 19-20).

555. **In sum**, to the extent that the Patent Owner asserts that this claim

requires monitoring and control functionality for load (resource usage) expansion or contraction across multiple servers in Vasell's **service gateway system** and that Vasell does not explicitly disclose this, in my opinion, Vasell's teachings in view of Rellermeyer's teachings of load balancing and elastic scalability disclose a load controller (a **topology manager** as part of Vasell's **remote management** functionality controlling Vasell's DRS functionality) adapted to monitor loads on adapted to monitor loads on at least one of the plurality of network device applications and at least one of the plurality of cloud applications and effect change in accordance with thresholds received from the application management portal (adapted to monitor, dynamically allocate, and control **service gateway system** resources (e.g., "CPU time, memory, persistent storage, and the like"), including load expansion/contraction across the plurality of **local servers** and **remote (cloud) servers** in the **service gateway system** "based upon a predetermined scheme of maximum allowed or minimum required resources" (including "overloaded" and "over-utilization" thresholds). *Id.*

5. **Claim 8: The system of claim 3 further comprising: a distributed notification service wherein notice of a predetermined event is sent from at least one of the plurality of network device applications to at least one of the plurality of cloud applications.**

556. Vasell discloses this limitation.

557. Vasell discloses "**cell manager**... also uses main gates... to add and

remove gate listeners”, where:

Listeners are entities, or classes, that have been registered to **receive event notifications** from other classes **upon the occurrence or pending occurrence of a specified event**. Whenever an event of the specified type occurs, a predetermined operation is invoked within the listener. For example, the occurrence of an alarm indication within **the fire alarm service application** of a household, could invoke a listener within **an audio system control service application** to shut down the household’s audio system so the fire alarm could be heard.

EX1004 (Vasell Patent), 20:22-23, 19:29-38, 19:21-29 (“[w]ithout the use of... proxies [*see my analysis of claims 14 and 15 (§§IX.B.5-IX.B.6 above)*], it would be impossible to send a listener across one of the gates”), 12:67-13:12.

558. A POSITA would have understood that part of the secure communications over the virtual fabric between the distributed **service application** “**portions**”/**components** would include these generated “event notifications... upon the occurrence... of a specified event.” *Id.*

559. **In sum**, in my opinion, a POSITA would have understood that, although it uses different words, Vasell’s disclosure of controlled implementation of “listeners” in the **service gateway system**, using “proxies” (*see claims 14 and 15 (§§IX.B.5-IX.B.6 above)*), to securely and “transparent[ly]” send generated “event notifications... upon the occurrence... of a specified event” between the respective underlying Java software “portions”/components of Vasell’s distributed **service**

**applications**, is the same as the '823 Patent's description of *a distributed notification service wherein notice of a particular event is sent from at least one of the plurality of network device applications to at least one of the plurality of cloud applications*. *Id.*; see also the Summary of the '823 Patent §VII.A above, (¶268 above, ¶270 above) (*citing* EX1001 ('823 Patent), 21:62-22:7, Table 8 (Use cases #5, #14, #67)).

**6. Claim 18: The system of claim 1, wherein the at least one programmable network device and the programmable cloud device are adapted to verify the authenticity and integrity of updates to the plurality of network device applications and the plurality of cloud applications.**

560. Vasell alone, or in view of Hall, discloses the limitations of claim 18.

561. For example, Vasell discloses “upgrade software” as a “**remote ... management**” operation, and describes an example of how the “**cell manager**” and “**cell table**” enable the implementation of services—“[e]ach service ... implemented as software **service applications**”—by the **service gateway system** to “continue undisturbed while [**service application**] configuration updates take place” [*updates to the plurality of network device applications and the plurality of cloud applications*]. EX1004 (Vasell Patent), 5:16-26, 22:19-30, 22:6-18, FIG. 6, 12:11-29, 2:56-59 (“It is a further purpose of the **service gateway system** that all configurations... be either locally or remotely downloadable.”).

562. Vasell 2<sup>nd</sup> Provisional provides details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“At the architecture level, the network operator has a key role, by

assuming the responsibility for maintenance and upgrades from the end-user.”), 6 (“system software upgrades”).

563. Vasell discloses that authentication of updates to distributed Java **service applications** is important due to, as discussed in my analysis of claim elements [1.4] above (§IX.B.1 above) and [3.2] above (§IX.C.1 above), the “open” nature of the **service gateway system**. Specifically, Vasell discloses that “[s]ervice gateways according to the present invention are preferably **open** platforms to allow for the development of **service applications** by **third party developers**” using “standard communication protocols and programming languages”, “service gateways according to the present invention are intended to be horizontal in nature so that a plurality of service providers can share the same platform infrastructure”, and “service gateway compatibility potentially opens **third-party markets** for... services.” EX1004 (Vasell Patent), 3:10-19, 5:30-33, 10:18-23, 2:60-66 (“[t]he **service gateway system** according to the present invention facilitates the development... of services... [by] **different services providers**” where “[e]ach service... [is] implemented as software **service applications**.”), 13:37-43.

564. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 4 (“*Development environment for **application software***: The development environment must follow the ‘write-once, run-everywhere’ maxim and should be based on Java standards. New **applications** will interact with the **e-service**

infrastructure through Java application program interfaces (API) that comply with mainstream Java development. By leveraging the Java development, the application software environment can be taken to a higher level of abstraction, allowing nonspecialists to develop service applications more easily.”), 8 (“*Development environment*: Boxlets are created using a standard Java development environment. For instance, the Java development kit (JDK) from Sun Microsystems can be used as well as other development environments. The only parts that are specific to boxlet development are the libraries that contain APIs for the main services and system services layers.”), 9 (identifying the disclosed “new e-service infrastructure” as “an open platform that can be used by several independent service providers”).

565. Moreover, Vasell discloses that, because “the service gateway system is accessed by and shared with multiple service providers and service gateway users”, “mechanisms for authentication... are integrated within the service gateway system design”, and “various authorization and identity verification techniques, etc. [are] to be provided so that the implementation... associated with these [connectivity-based] services is robustly defended against intrusion and manipulation.” *Id.*; EX1004 (Vasell Patent), 4:52-65, 11:59-12:10, 15:46-47.

566. Vasell 2<sup>nd</sup> Provisional adds details. EX1016 (Vasell 2<sup>nd</sup> Provisional), 3 (“Consumer Requirements of E-Services: ... *Security*: Consumers do not tolerate invasion of privacy or hacking into mission-critical services by strangers or

neighbors.”), 4 (“Service Provider Requirements of an **E-Service Infrastructure**: ... *Integrity*: Different service providers must be allowed to share the infrastructure without adversely affecting each other’s services.”), 6 (“[T]he fact that the system **will** be shared by multiple service providers, make security a major issue. Mechanisms for **authentication and authorization must** be well integrated into the *e-box* system design.”).

567. Unsurprisingly, Vasell does not disclose the details of its **service gateway system local** and **remote** servers [*programmable network device* and *programmable cloud device* respectively] (*see* my analysis of claim elements [1.1] above (§IX.B.1 above) and [1.2] above (§IX.B.1 above)), respectively implementing well-known “authentication”, “authorization”, and/or “identity verification” security techniques for its intended “open” development environment of received distributed **service application software** by “different service providers”, “third party developers”, “non-specialists” and “third party markets for... services”, especially where Vasell discloses that such distributed **service applications** are “Java **applications**” implemented in “a Java runtime in the form of a Java virtual machine (JVM)” environment for which, as discussed in the Technology Background §VI.D.3 above (Java Virtual Machines and Java Applications) and §VI.E above (Security for Software Implemented and Downloaded Over the Internet Including Digital Signature Implementations), such security techniques were widely

implemented by recipient devices. *Id.*

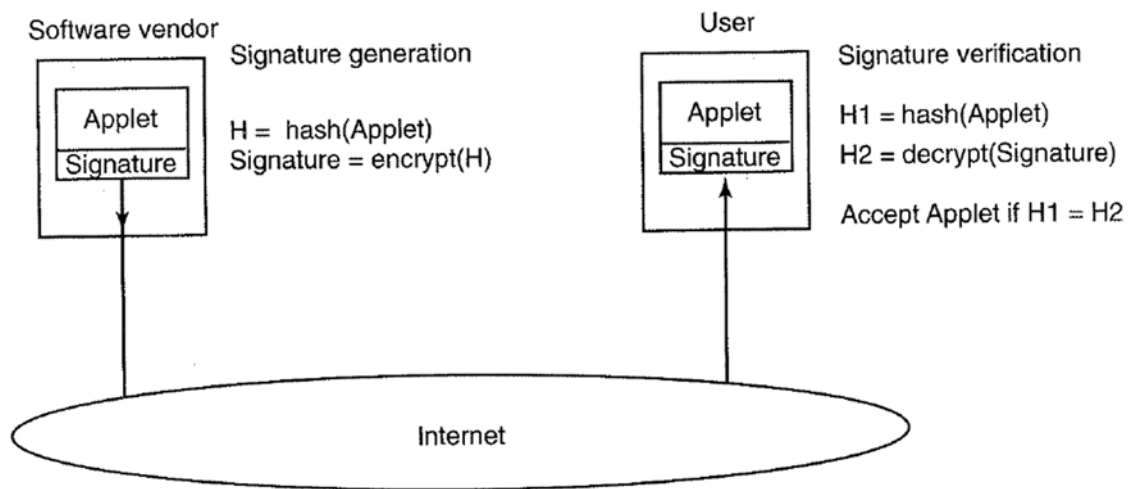
568. I discuss these details in my Technology Background. *See* my Technology Background §VI.D.3 above (Java Virtual Machines and Java Applications) and §VI.E above (Security for Software Implemented and Downloaded Over the Internet Including Digital Signature Implementations), ¶¶133-147 above (*citing* EX1034 (Cole), 305 (The “Java sandbox” was well-known to be “composed of ... - **Code stores** – The sites that the **applications** are physically stored on prior to execution on the host; - **Certificates** – Used to sign **code** to convey trust to a user that you are the developer of an **application**; – **Key Stores** – Files that contain the certificates... Key Stores are queried to identify who signed the **application code**.”) (original emphasis).

569. Tanenbaum provides details of Java’s security. *Id.* (*citing* EX1044 (Tanenbaum 2001), 643-644 (“Java **programs** are compiled to an intermediate binary code called **JVM (Java Virtual Machine) byte code** ... In the Java model, **applets** sent over the Internet for remote execution are JVM **programs**. When an **applet** arrives, it is run through a JVM byte code verifier that checks if the **applet** obeys certain rules ... **JDK (Java Development Kit)** ... 1.2 provides a configurable fine-grain security policy that applies to all **applets** ... Each **applet** is characterized by two things: where it came from and who signed it. Where it came from is its URL; who signed it is which private key was used for the signature. Each user can create

a security policy consisting of a list of rules. A rule may list a URL, a signer, an object, and an action that the **applet** may perform on the object if the applet's URL and signer match the rule. ... To verify the signatures, [the recipient] must either have the necessary public keys on [its] disk or must acquire them dynamically, for example in the form of a certificate signed by a company [it] trusts and whose public key [it] already has.”) (original emphasis), Figure 9-20 (which I have reproduced below)).

570. Tanenbaum describes Java's code signing. *Id.* (citing EX1044 (Tanenbaum 2001), 641-642 (“**Code** signing is based on public-key cryptography. An **applet** vendor, typically a software company, generates a (public key, private key) pair, making the former public and zealously guarding the latter. To sign an **applet**, the vendor first computes a hash function of the **applet** to get a 128-bit or 160-bit number, depending on whether MD5 or SHA is used. It then signs the hash value by encrypting it with its private key (actually, decrypting it using the notation of Fig. 9-3). This signature accompanies the **applet** wherever it goes. When the user gets the **applet**, [it] computes the hash function itself. It then decrypts the accompanying signature using the vendor's public key and compares what the vendor claims the hash function is with what ... itself computed. If they agree, the **applet** is accepted as genuine. Otherwise it is rejected as a forgery. The mathematics involved makes it exceedingly difficult for anyone to tamper with the **applet** in such

a way as its hash function will match the hash function that is obtained by decrypting the genuine signature. It is equally difficult to generate a new false signature that matches without having the private key. The process of signing and verifying is illustrated in Fig. 9-20.”).



**Figure 9-20.** How code signing works.

571. My Technology overview provides more details. See §VI.B.1 above (Technology Overview – Hash Functions and Their Uses) (citing EX1057 (Microsoft Computer Dictionary), 432, 131, 145, 388-389)).

572. As also identified in my Technology overview, the OSGi itself had described Java security. §VI.E above (Technology Overview – Security for Software Implemented and Downloaded Over the Internet Including Digital Signature Implementations) (citing EX1050 (“OSGi 2000 Overview”), 8-10 (“The first OSGi specifications will concentrate on **Java APIs** ... A **bundle** is a **Java archive (JAR) file** ... The JAR manifest file format is extended to contain the

additional information about the **bundle** and **the standard JAR signing technique** is used to ensure the integrity and authenticity of downloaded **bundles**.”)).

573. Lastly, Hall also confirmed Java’s security mode. *Id.* (citing EX1009 (Hall), 457 (“Digitally signed **bundles** can ... help you do two things: ■ Authenticate the provider of a **bundle** ■ Ensure that **bundle** content hasn’t been modified.”)).

574. Indeed, as I described in my Technology Background §VI.E above (Technology Overview – Security for Software Implemented and Downloaded Over the Internet Including Digital Signature Implementations, ¶¶136-147 above), a standardized security scheme that was well-known to POSITAs and widely implemented by application developers and implementers for JVM-hosted **applications**, in the precise development environment described by Vasell (including Java **service application** developers with varying and unknown trustworthiness levels), was to require each developer to “digitally sign” each of its respectively developed **software applications** (and each of its respective upgrades to such **software**), or a respective “hash” of each application/update (computed using standardized algorithms (*e.g.*, SHA, MD5, etc.)), using a unique private key issued to it by a trusted authority, so that, prior to installation of each application (or upgrade (sometimes called an “update” or “patch”) of such application), a management agent (and each “JVM security manager” of the JVMs in which the **bundles** are deployed) would use the corresponding unique public key (typically

identified in an X.509 certificate of the developer) to verify that the respective developer is authentic and the respective software (upgrade to such software) is unmodified. *Id.*

575. Garfinkel provides additional details. *Id.* (citing EX1045 (Garfinkel), 169-172 (“Code signing is a technique for signing executable programs with digital signatures. Code signing is designed to improve the reliability of software that's distributed over the Internet. It is meant to provide a system for trusting downloaded code and reducing the impact of malicious programs ... Code signing is supposed to bring the assurance of shrink-wrapped software to the world of software that's distributed electronically. It does this by adding two things to an executable: • A digital signature that signs the executable with a secret key. • A digital certificate, which contains the corresponding public key, the name of the person or organization to whom that key belongs, and a digital signature signed by a recognized certification authority. ... Ideally, all code signatures should be verified **when each piece of code is downloaded as well as before each time it is run.**”) (bold emphasis added), 265-266 (“**Before you install** any [software] patch, be sure that the patch is an authentic patch provided by your vendor ... [by] checking [its] digital signatures and cryptographic checksum to determine the authenticity and integrity of a patch **before you install it.**”) (bold emphasis added)).

576. Thus, in my opinion, a POSITA would have understood that Vasell

discloses the limitations in claim 18 based on Vasell's disclosed implementation of services in the **service gateway system** as distributed "Java **applications**", with each "portion"/component of each **application** hosted on a respective Java Virtual Machine (JVM) running on the respective OSs of the **local** and **remote (cloud)** servers of Vasell's **service gateway system**, and Vasell's concerns regarding the authentication of any upgrades to such **applications**. *Id.*

577. **In sum**, in my opinion, a POSITA would have understood that Vasell discloses wherein the at least one programmable network device (**service platform server 22**, *see also* my analysis of claim element [1.1] above (§IX.B.1 above)) and the programmable cloud device ("**service provider equipment 34**", "**network operator server 24**", *see also* my analysis of claim element [1.2] above (§IX.B.1 above)) are adapted to verify the authenticity and integrity of updates to the plurality of network device applications and the plurality of cloud applications (wherein, as part of the "upgrades" operation of the plurality of distributed, Java **service applications** on the **local** and **remote (cloud)** servers in Vasell's **service gateway system**, prior to respective installation of each Java **application/upgrade** on the corresponding **server**, the JVM(s) respectively running on the host OS of Vasell's **local** and **remote (cloud)** servers respectively use the respective unique public key corresponding to the developer of the Java **application/upgrade** (typically identified in an X.509 certificate of the developer) to verify that the respective developer is

authentic and the respective Java **application**/upgrade is unmodified).

578. Moreover, to the extent that the Patent Owner asserts that Vasell does not explicitly disclose this limitation to POSITAs, in my opinion, it would have been obvious to POSITAs to modify Vasell’s teachings with Hall’s teachings of requiring OSGi-application-developers to “digitally sign” their respectively developed distributed **application components** (**bundles** and/or application deployment packages including such **bundles**), including updated versions of such **components** of such distributed **applications**, as I discussed in the Motivation to Combine §IX.A above, to achieve, and with a reasonable expectation of achieving, the benefits expressly disclosed by Vasell (*see, e.g.*, ¶¶565-566 above) and other well-known benefits (*e.g.*, rendering each distributed **application** “portion”/component (and upgrade of the same) “tamperproof”, authenticating the provider of such upgrades (by verifying that the signer has access to the unique private key), ensuring that the content of such upgrades is unmodified, determine the application-cell-specific-access-levels granted by Vasell’s **management services** to each such **application** “portion”/component (and upgrade of the same) based on the respective developer’s identity (signature), and to control the **application** “portion[’s]” (and upgrade’s) access to operations, information, services and **server** resources). *Id.*; *see* §IX.A above (Motivation to Combine) *citing* §VIII.A.3 above (Overview of Hall section) *citing* EX1009 (Hall), 457-463, 331-334, 439-448, 471-476, 179)).

579. **In sum**, in my opinion, to the extent that the Patent Owner asserts that Vasell does not explicitly disclose this limitation to POSITAs, a POSITA would have understood that Vasell, in view of Hall’s teachings of requiring OSGi-application-developers to “digitally sign” their respectively developed distributed **application components** (**bundles** and/or application deployment packages including such **bundles**), including updated versions of such **components**, discloses *the programmable network device and the programmable cloud device* (the respective JVM(s) running on the host OS of the recipient **server** (of each such distributed **application** “portion”/component (and upgrades)) of the **service gateway system**), are adapted to verify the authenticity and integrity of updates to the plurality of network device applications and the plurality of cloud applications (wherein, as part of its distributed Java **service application software** “upgrades” operation, the respective JVM(s) running on the host OS of the recipient **server** of each such distributed **application** “portion”/component (and upgrades), prior to respective installation of each Java **application**/upgrade, uses the respective unique public key corresponding to the developer of the Java **application**/upgrade (typically identified in an X.509 certificate of the developer) to perform standard Java digital signature matching to verify that the respective developer is authentic and the respective Java **application**/upgrade is unmodified, determine the application-cell-specific-access-levels granted by Vasell’s **management services** to each such **application**

“portion”/component (and upgrade) based on the respective developer’s identity (signature), and to control the application “portion[s]” (and upgrades’) access to operations, information, services and server resources).

## **X. CONCLUSION**

In my opinion, the challenged claims 1-5, 7-8, 12-13, 15 and 18-19 of the '823 Patent are invalid.

### **A. AVAILABILITY FOR CROSS-EXAMINATION**

In signing this declaration, I recognize that the declaration will be filed as evidence in a case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross examination in the case and that cross examination will take place within the United States. If cross examination is required of me, I will appear for cross examination within the United States during the time allotted for cross examination.

### **B. RIGHT TO SUPPLEMENT**

I reserve the right to supplement my opinions in the future to respond to any arguments that the Patent Owner may raise and to take into account new information as it becomes available to me.

### **C. JURAT**

I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1101 of Title 18 of the United States Code.

[SIGNATURE ON FOLLOWING PAGE]

Dated: February 14, 2025

A handwritten signature in black ink, appearing to read "Erez Zadok". The signature is written in a cursive, fluid style with some loops and flourishes.

By: \_\_\_\_\_

Dr. Erez Zadok